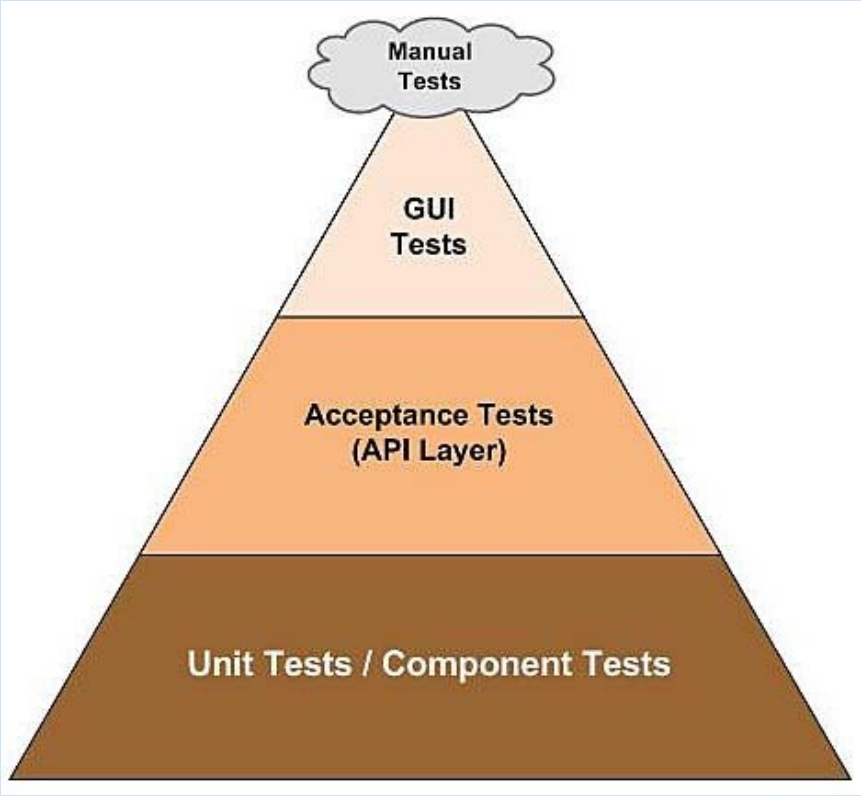
Agile Swift

Godfrey Nolan

RIIS LLC

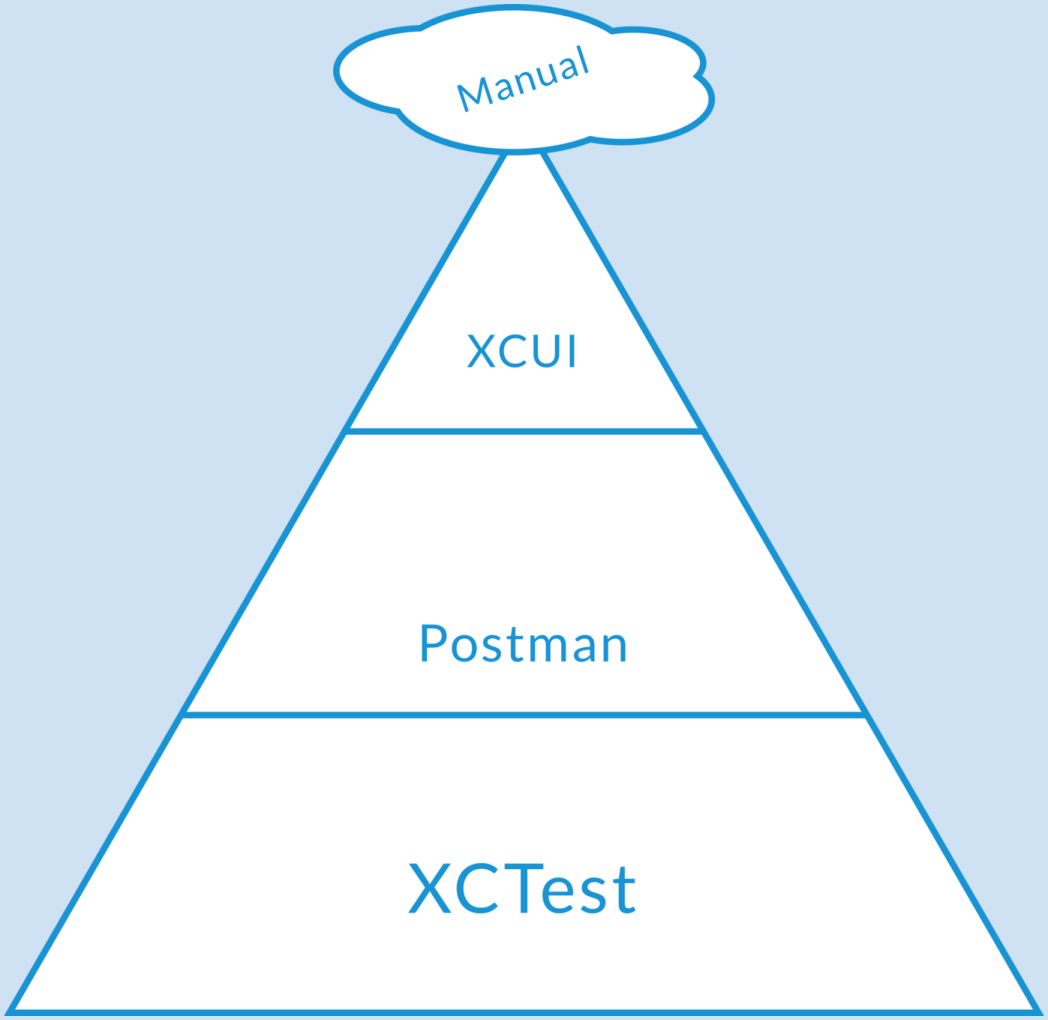
Agenda


Manual
Tests
GUI
Tests
Acceptance Tests
(API Layer)
Unit Tests / Component Tests

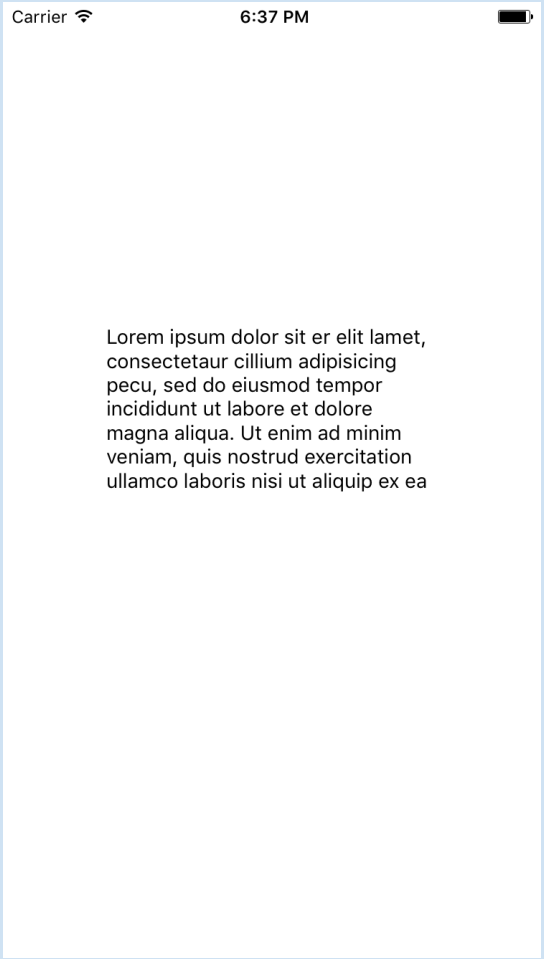
Agile Testing Benefits

- Catch more mistakes
- Confidently make more changes
- Built in regression testing
- Extend the life of your codebase
- Better predicability & reliability

Agile Swift



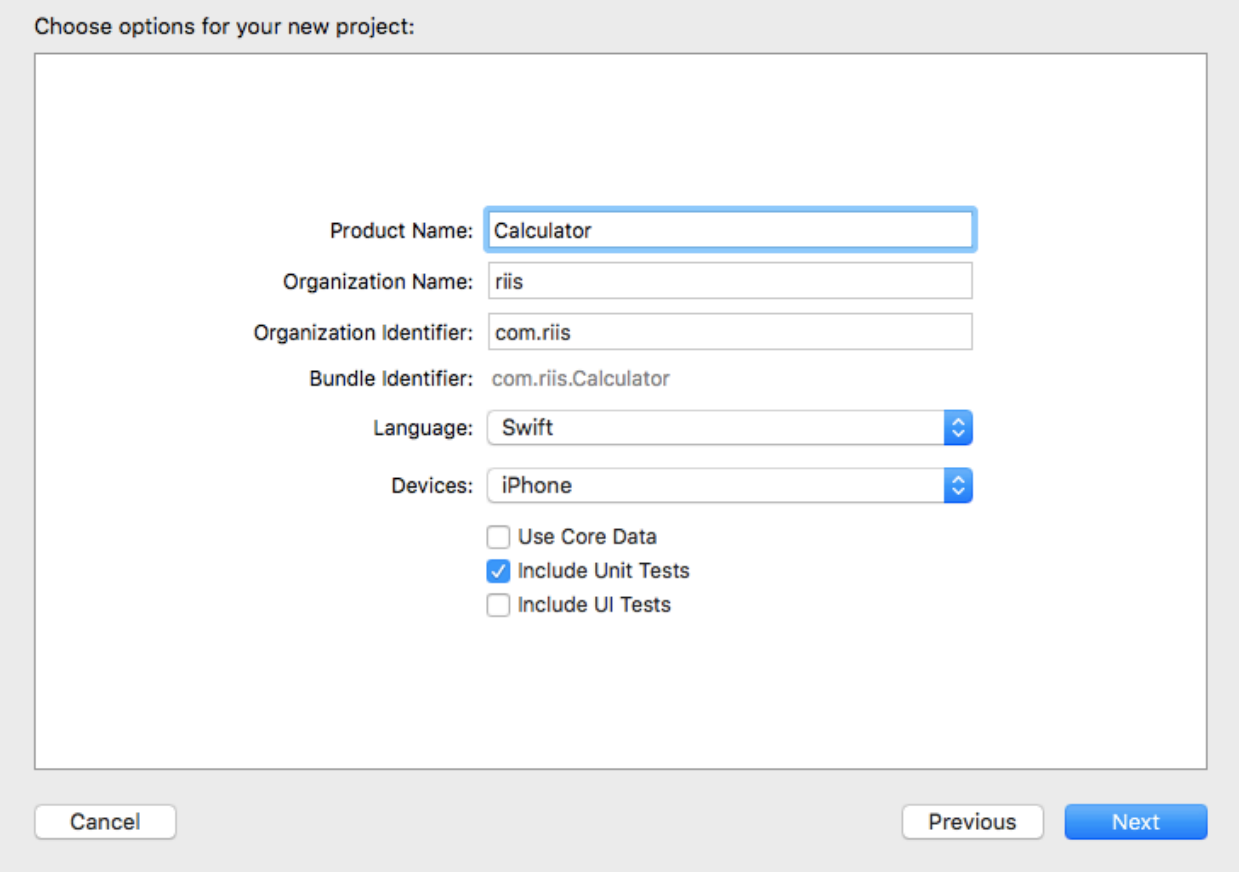
Hello World


Carrier
6:37 PM
Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea

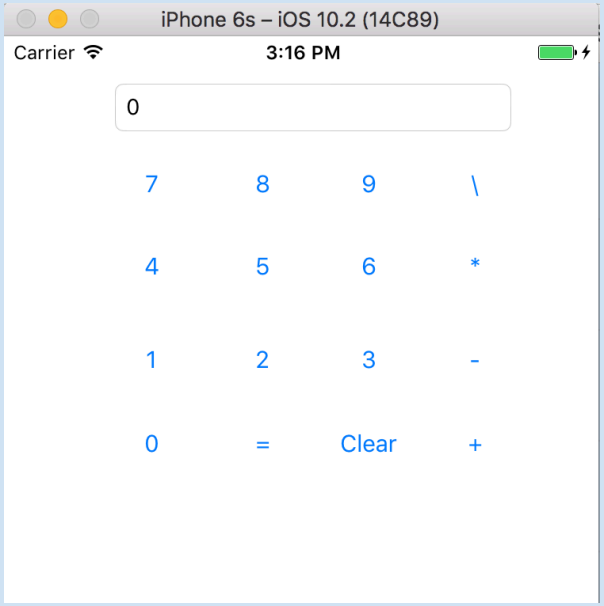
Lab 1 - HelloWorld

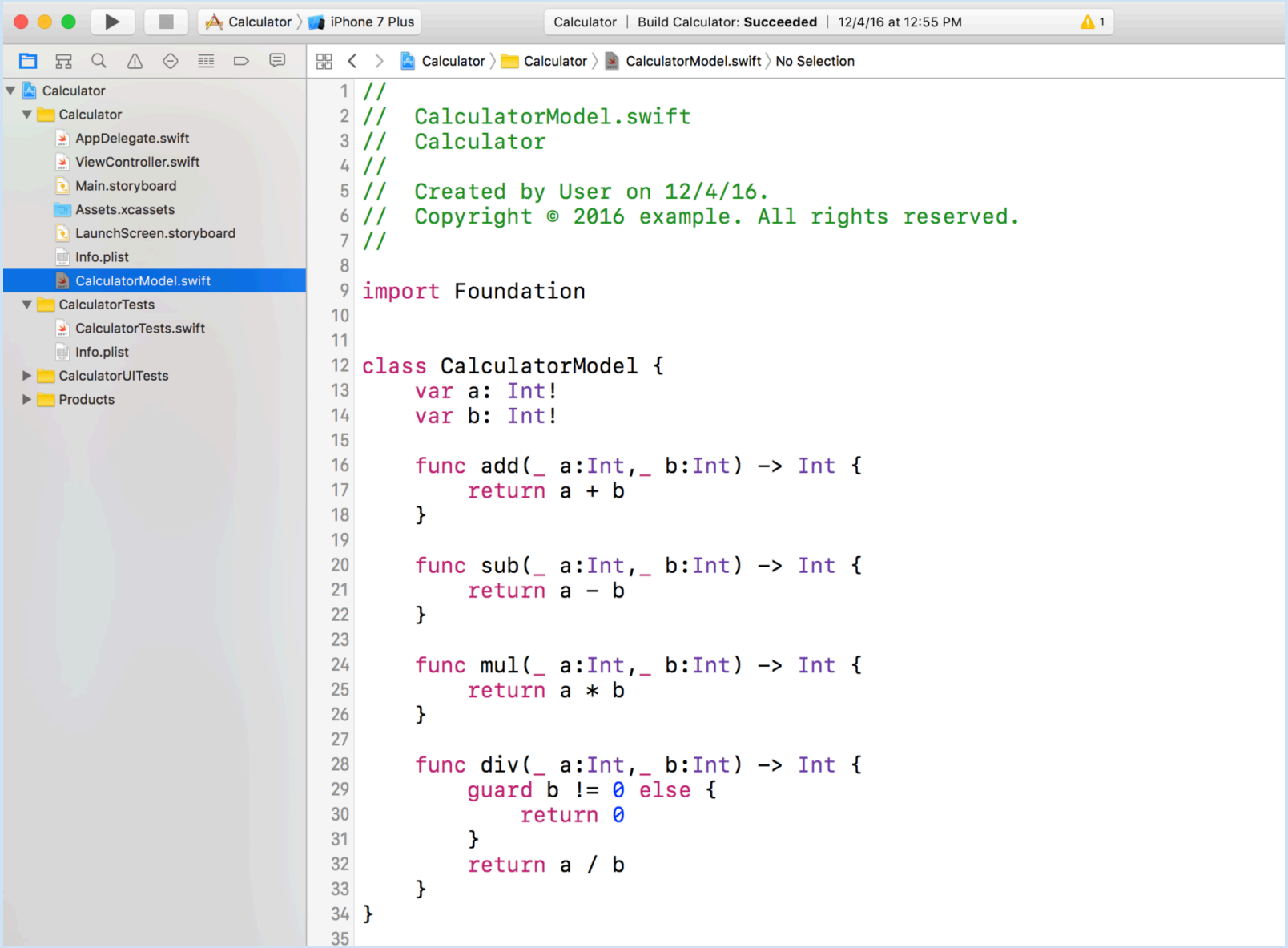
1. Open Xcode
2. Make sure Xcode is 8.x
3. Open Main.storyboard
4. Search for TextView in the Object Library
5. Drag over to Storyboard
6. Run

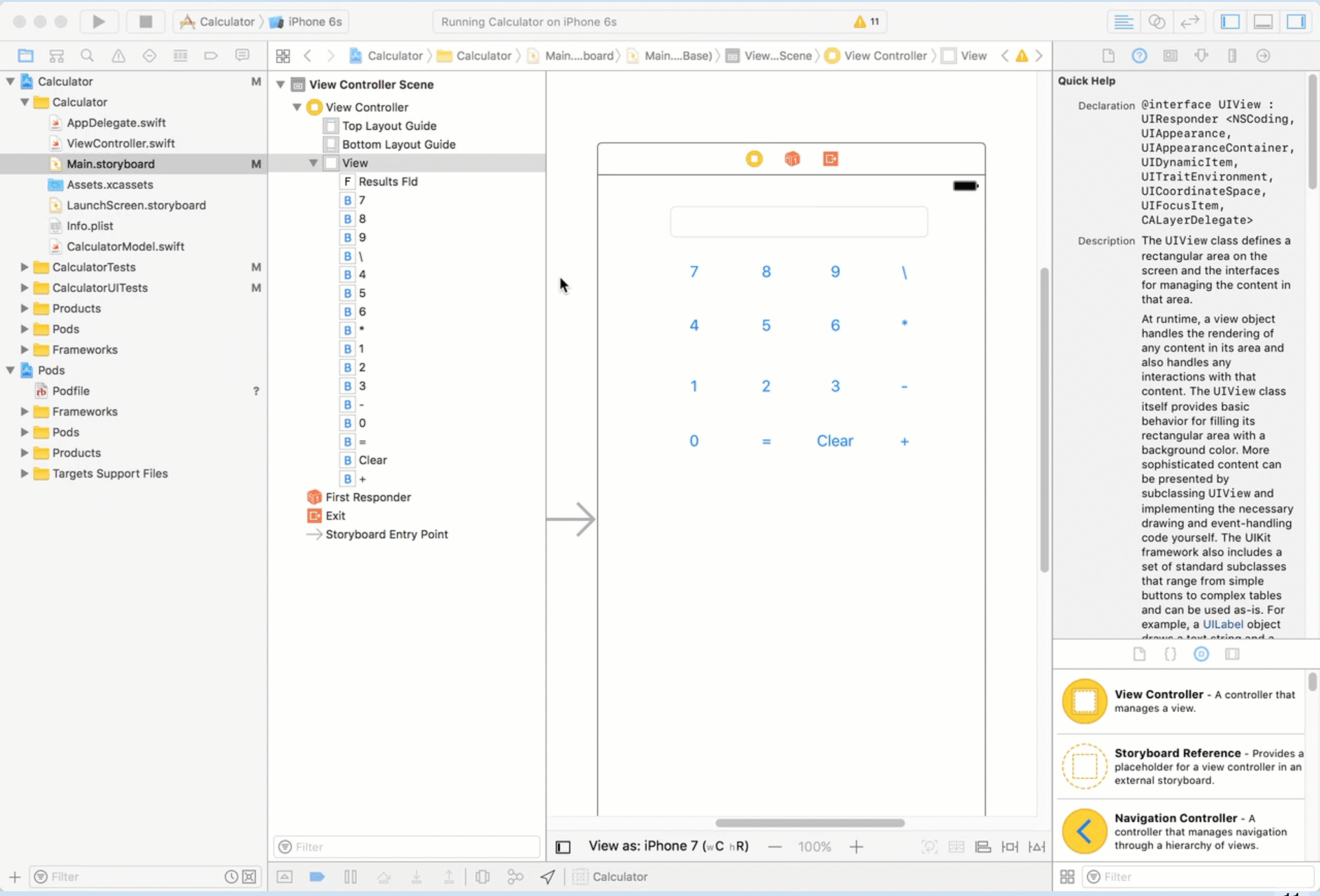
Swift Unit Test - XCTest



Swift Unit Test - XCTest




Calculator 〉 iPhone 7 Plus
Calculator | Build Calculator: Succeeded | 12/4/16 at 12:55 PM
Calculator 〉 Calculator 〉 CalculatorModel.swift 〉 No Selection
Calculator
Calculator
AppDelegate.swift
ViewController.swift
Main.storyboard
Assets.xcassets
LaunchScreen.storyboard
Info.plist
CalculatorModel.swift
CalculatorTests
CalculatorTests.swift
Info.plist
CalculatorUITests
Products
//
// CalculatorModel.swift
// Calculator
//
// Created by User on 12/4/16.
// Copyright © 2016 example. All rights reserved.
//
import Foundation
class CalculatorModel {
var a: Int!
var b: Int!
func add(_ a:Int,_ b:Int) -> Int {
return a + b
}
func sub(_ a:Int,_ b:Int) -> Int {
return a - b
}
func mul(_ a:Int,_ b:Int) -> Int {
return a * b
}
func div(_ a:Int,_ b:Int) -> Int {
guard b != 0 else {
return 0
}
return a / b
}
}


Calculator
iPhone 6s
Running Calculator on iPhone 6s
Calculator
Calculator
AppDelegate.swift
ViewController.swift
Main.storyboard
Assets.xcassets
LaunchScreen.storyboard
Info.plist
CalculatorModel.swift
CalculatorTests
CalculatorUITests
Products
Pods
Frameworks
Pods
Podfile
Frameworks
Pods
Products
Targets Support Files
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Guide
View
Results Fld
7
8
9
\
4
5
6
*
1
2
3
-
0
=
Clear
+
First Responder
Exit
Storyboard Entry Point
Main....board
Main....Base)
View...Scene
View Controller
View
7 8 9 \
4 5 6 *
1 2 3 -
0 = Clear +
View as: iPhone 7 (wC hR)
100%
Quick Help
Declaration
@interface UIView : UIResponder <NSCoding, UIAppearance, UIAppearanceContainer, UIDynamicItem, UITraitEnvironment, UICoordinateSpace, UIFocusItem, CALayerDelegate>
Description
The UIView class defines a rectangular area on the screen and the interfaces for managing the content in that area.
At runtime, a view object handles the rendering of any content in its area and also handles any interactions with that content. The UIView class itself provides basic behavior for filling its rectangular area with a background color. More sophisticated content can be presented by subclassing UIView and implementing the necessary drawing and event-handling code yourself. The UIKit framework also includes a set of standard subclasses that range from simple buttons to complex tables and can be used as-is. For example, a UILabel object
View Controller - A controller that manages a view.
Storyboard Reference - Provides a placeholder for a view controller in an external storyboard.
Navigation Controller - A controller that manages navigation through a hierarchy of views.
Calculator

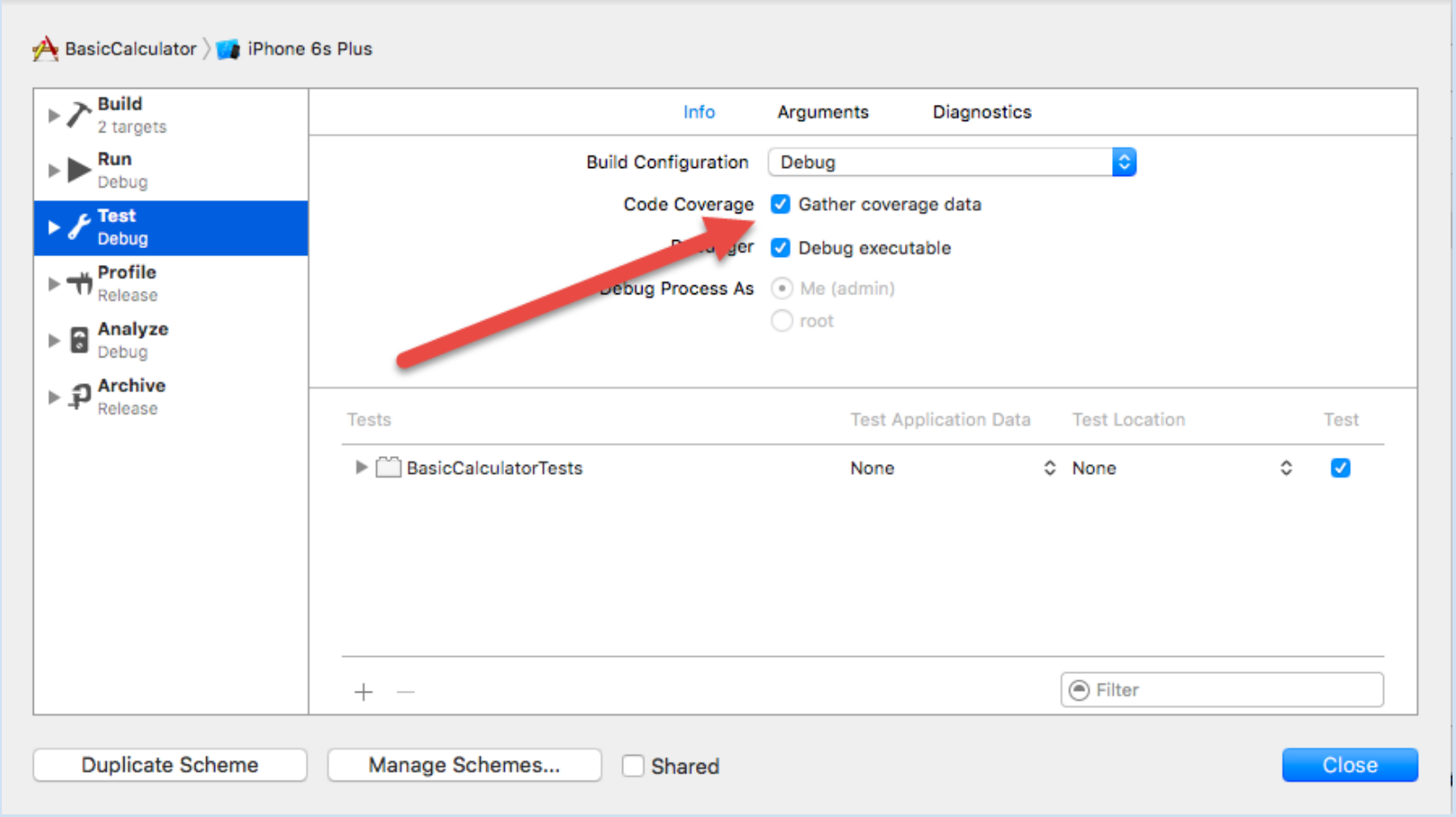
Lab 2 - Calculator

```
0. Download code from https://github.com/godfreynolan/CodeCraftsman
1. Open Calculator in iOS Folder
2. Click on Calculator Tests
3. Open CalculatorTests.swift
4. Run tests
5. Open Test Navigator
6. Run tests again
7. Open Report Navigator
8. View Logs->All Messages
9. Add more tests to CalculatorTests for sub, mul, div functions
```

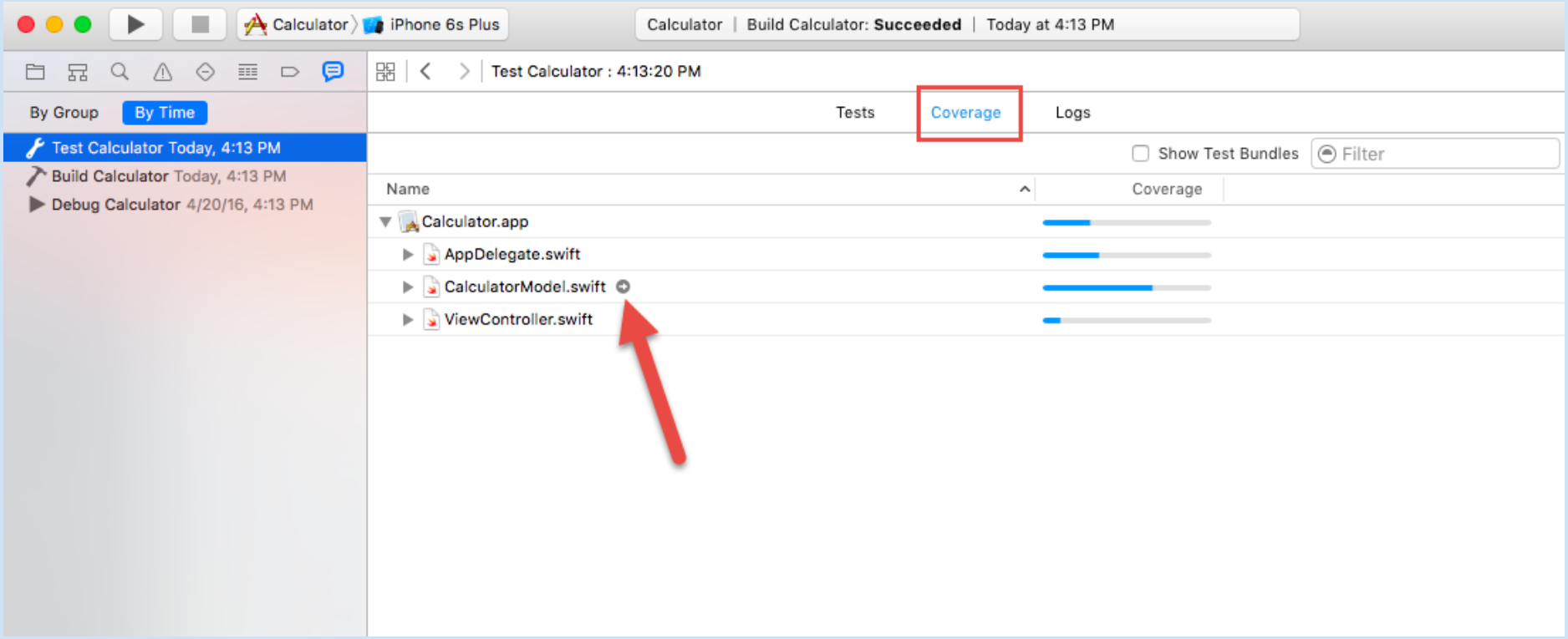
Unit Testing Assertions

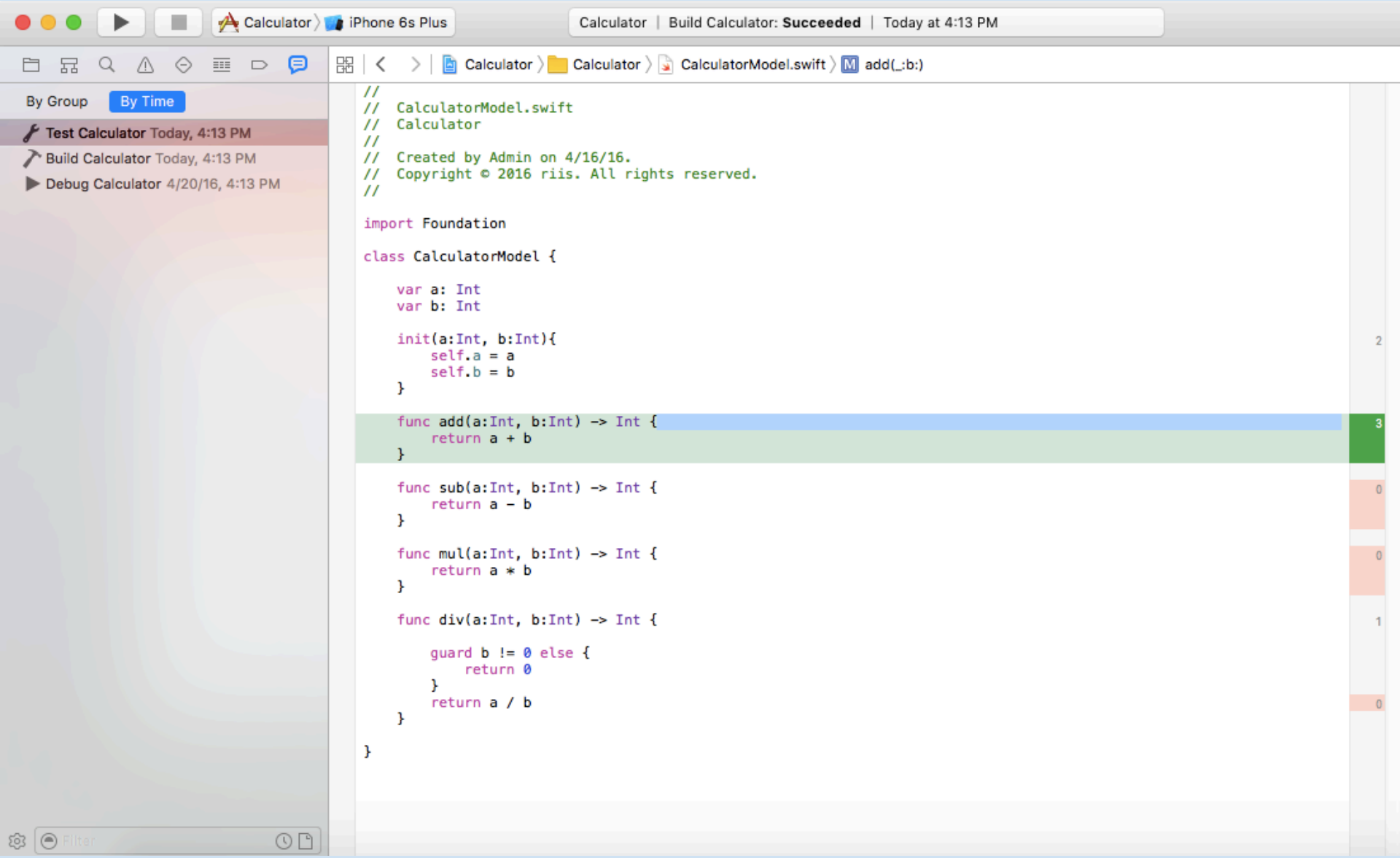
Assertion	Description
`XCTAssert`	Tests that two values are the same
`XCTAssertEqual`	Tests that two values are equal
`XCTAssertEqualWithAccuracy`	Tests that two floating point values (a,b) are equal within a tolerance of c
`XCTAssertFalse`	Tests if a Boolean condition is false
`XCTAssertTrue`	Tests if a Boolean condition is true
`XCTAssertGreaterThan`	Tests that one value is greater than the other
`XCTAssertGreaterThanOrEqual`	Tests that one value is greater or equal to the other
`XCTAssertLessThan`	Tests that one value is less than the other
`XCTAssertLessThanOrEqual`	Tests that one value is less than or equal to the other
`XCTAssertNil`	Tests that an object is nil
`XCTAssertNotEqual`	Tests that two values are not equal
`XCTAssertNotEqualWithAccuracy`	Tests that two floating point values (a,b) are not equal within a tolerance of c
`XCTAssertNotNil`	Tests that an object is not nil

Unit Testing Code Coverage



Unit Testing Code Coverage

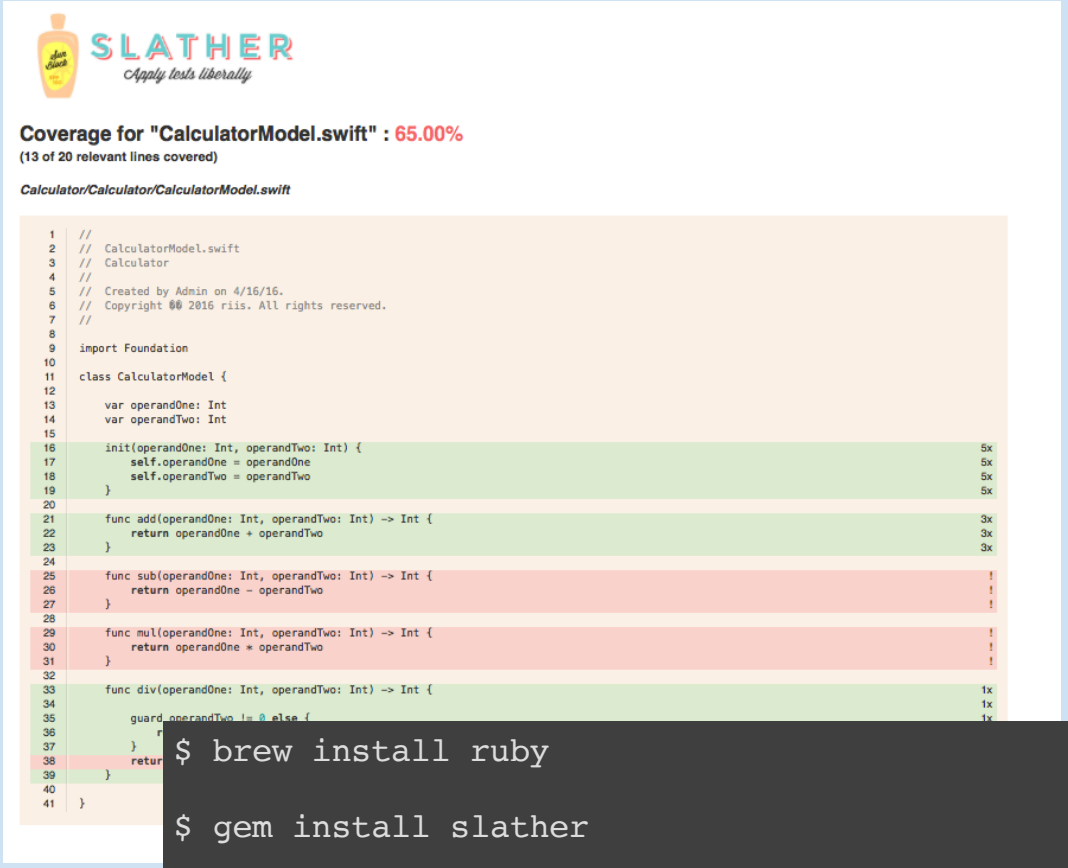



Calculator 〉 iPhone 6s Plus
Calculator | Build Calculator: Succeeded | Today at 4:13 PM
Calculator 〉 Calculator 〉 CalculatorModel.swift 〉 M add(_:b:)
By Group
By Time
Test Calculator Today, 4:13 PM
Build Calculator Today, 4:13 PM
Debug Calculator 4/20/16, 4:13 PM
//
// CalculatorModel.swift
// Calculator
//
// Created by Admin on 4/16/16.
// Copyright © 2016 riis. All rights reserved.
//
import Foundation
class CalculatorModel {
var a: Int
var b: Int
init(a:Int, b:Int){
self.a = a
self.b = b
}
func add(a:Int, b:Int) -> Int {
return a + b
}
func sub(a:Int, b:Int) -> Int {
return a - b
}
func mul(a:Int, b:Int) -> Int {
return a * b
}
func div(a:Int, b:Int) -> Int {
guard b != 0 else {
return 0
}
return a / b
}
}
2
3
0
0
1
0

Code Coverage Lab

1. Open Calculator in iOS Folder
2. Enable Code Coverage
3. Run tests
4. View Code coverage results

Unit Testing Code Coverage




```
$ brew install ruby

$ gem install slather

$ slather coverage --html --scheme XcodeSchemeName path/to/project.xcodeproj

$ slather coverage --html --scheme Calculator Calculator/Calculator.xcodeproj
```

Code Coverage Lab #2

```
1. Install slather
    brew install ruby
    gem install slather
2. From the dir above Calculator run
    slather coverage --html --scheme Calculator Calculator/Calculator.xcodeproj
3. Find the html output
```

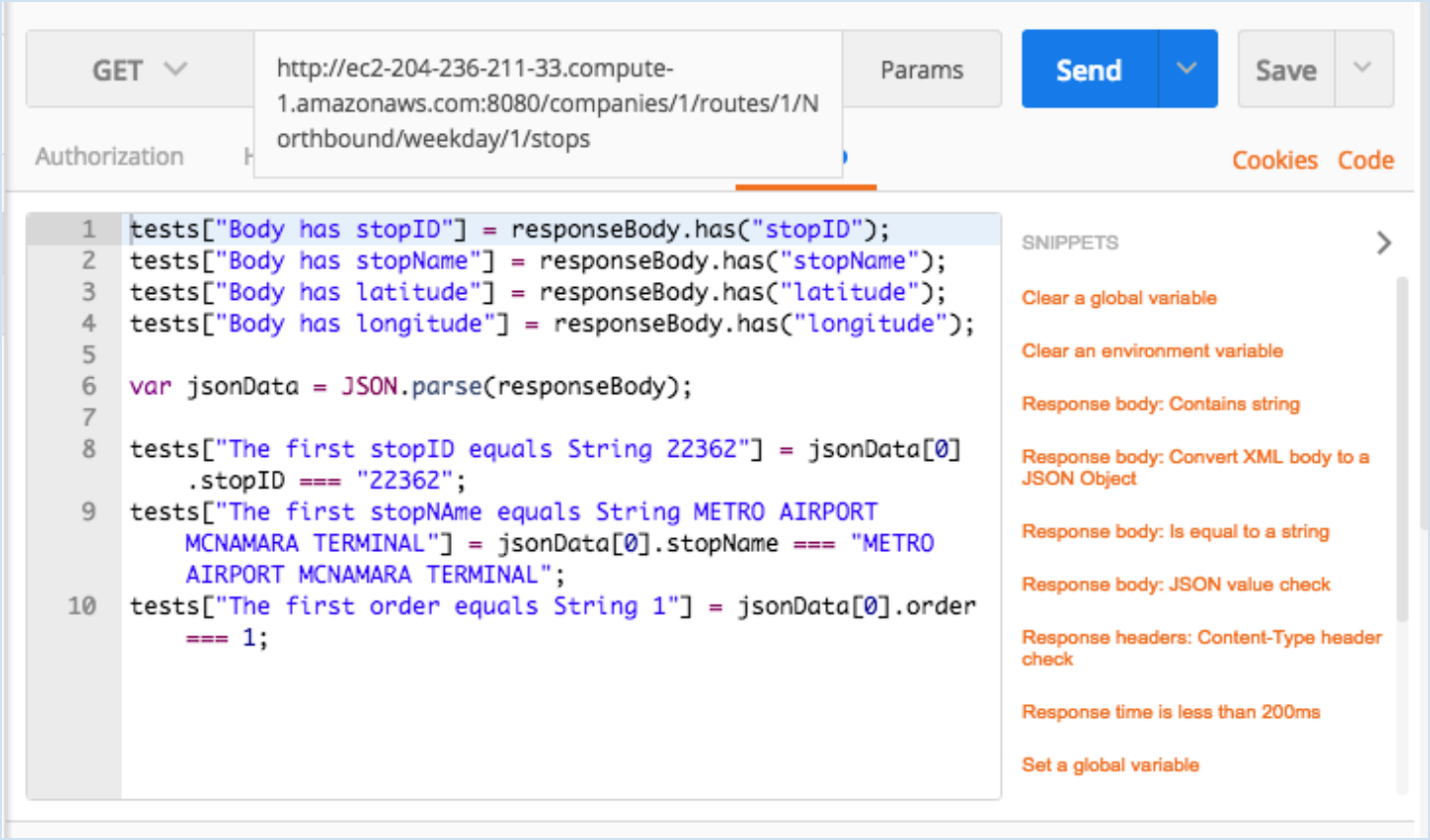
Swift on other platforms

```
RIIS-MBPRO-13:HelloWorld2 User$ swift package init
Creating library package: HelloWorld2
Creating Package.swift
Creating .gitignore
Creating Sources/
Creating Sources/HelloWorld2.swift
Creating Tests/
Creating Tests/LinuxMain.swift
Creating Tests/HelloWorld2Tests/
Creating Tests/HelloWorld2Tests/HelloWorld2Tests.swift
RIIS-MBPRO-13:HelloWorld2 User$
```

Swift on other platforms

```
RIIS-MBPRO-13:HelloWorld2 User$ swift test
Compile Swift Module 'HelloWorld2' (1 sources)
Compile Swift Module 'HelloWorld2Tests' (1 sources)
Linking ./.build/debug/HelloWorld2PackageTests.xctest/Contents/MacOS/HelloWorld2PackageTests
Test Suite 'All tests' started at 2017-04-07 18:45:50.743
Test Suite 'HelloWorld2PackageTests.xctest' started at 2017-04-07 18:45:50.744
Test Suite 'HelloWorld2Tests' started at 2017-04-07 18:45:50.744
Test Case '-[HelloWorld2Tests.HelloWorld2Tests testExample]' started.
Test Case '-[HelloWorld2Tests.HelloWorld2Tests testExample]' passed (0.002 seconds).
Test Suite 'HelloWorld2Tests' passed at 2017-04-07 18:45:50.747.
         Executed 1 test, with 0 failures (0 unexpected) in 0.002 (0.002) seconds
Test Suite 'HelloWorld2PackageTests.xctest' passed at 2017-04-07 18:45:50.747.
         Executed 1 test, with 0 failures (0 unexpected) in 0.002 (0.003) seconds
Test Suite 'All tests' passed at 2017-04-07 18:45:50.747.
         Executed 1 test, with 0 failures (0 unexpected) in 0.002 (0.004) seconds
```

API Testing - Postman



API Testing - Newman

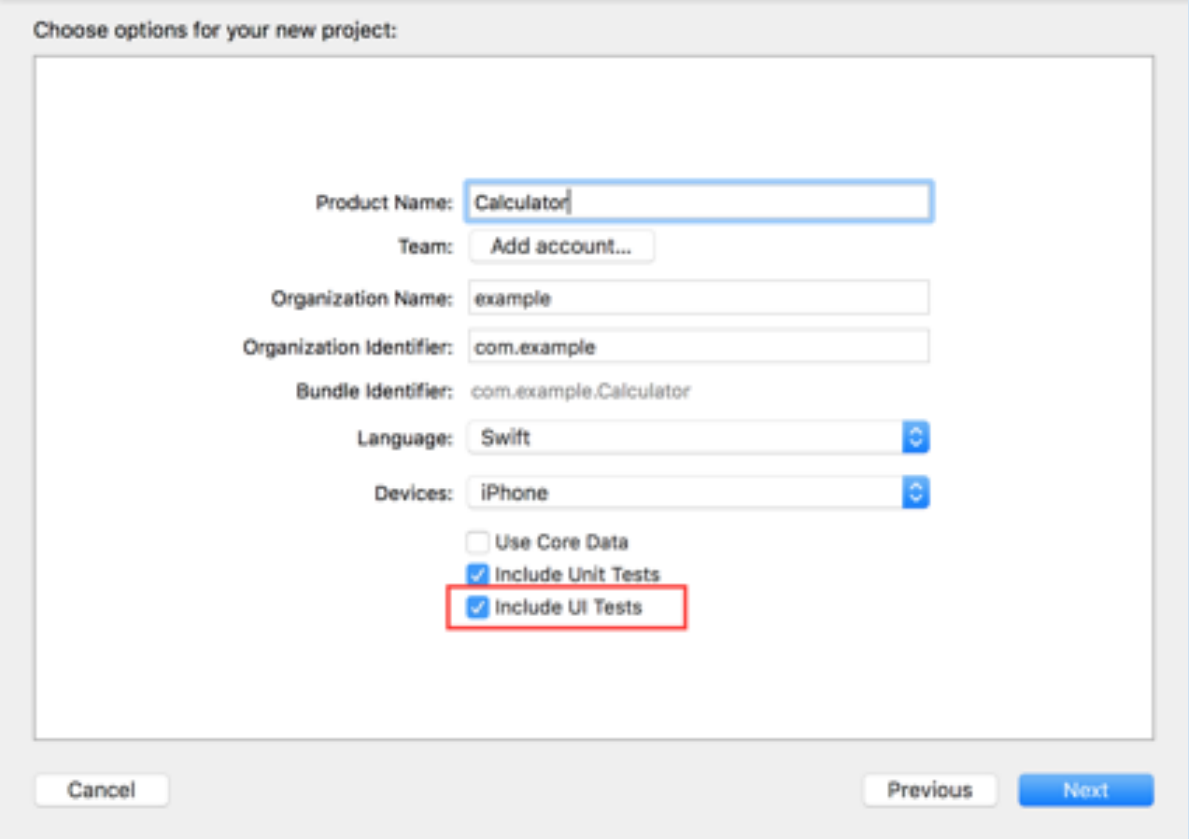
```
RIIS-MBPRO-13:Android User$ newman run ETAJson.postman_collection.json
newman

ETAJson

→ http://ec2-204-236-211-33.compute-1.amazonaws.com:8080/companies/1/routes/1/Northbound/weekday/1/stops
  GET http://ec2-204-236-211-33.compute-1.amazonaws.com:8080/companies/1/routes/1/Northbound/weekday/1/stops [200 OK, 1.74KB,
 103ms]
  ✓  Body has stopID
  ✓  Body has stopName
  ✓  Body has latitude
  ✓  Body has longitude
  ✓  The first stopID equals String 22362
  ✓  The first stopNAme equals String METRO AIRPORT MCNAMARA TERMINAL
  ✓  The first order equals String 1
```

	executed	failed
iterations	1	0
requests	1	0
test-scripts	1	0
prerequest-scripts	0	0
assertions	7	0
total run duration: 152ms		
total data received: 1.59KB (approx)		
average response time: 103ms		

API Testing - Newman

```
1. Download nodejs from https://nodejs.org/en/download/package-manager
2. Install newman
    npm install newman --global;
3. Run the tests
    newman run ETAJson.postman_collection.json
```

Swift GUI Testing - XCUI



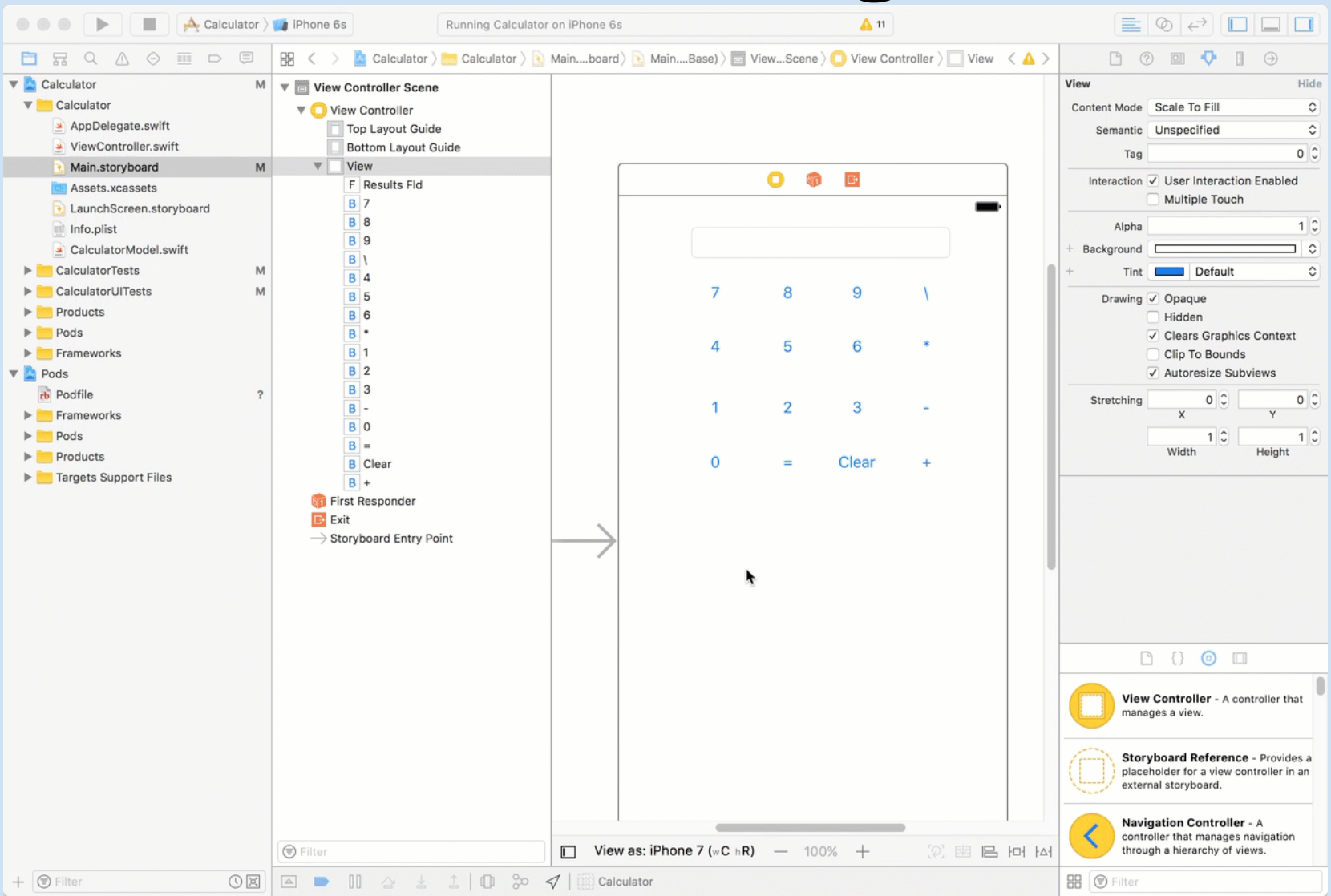
Swift GUI Testing - XCUI

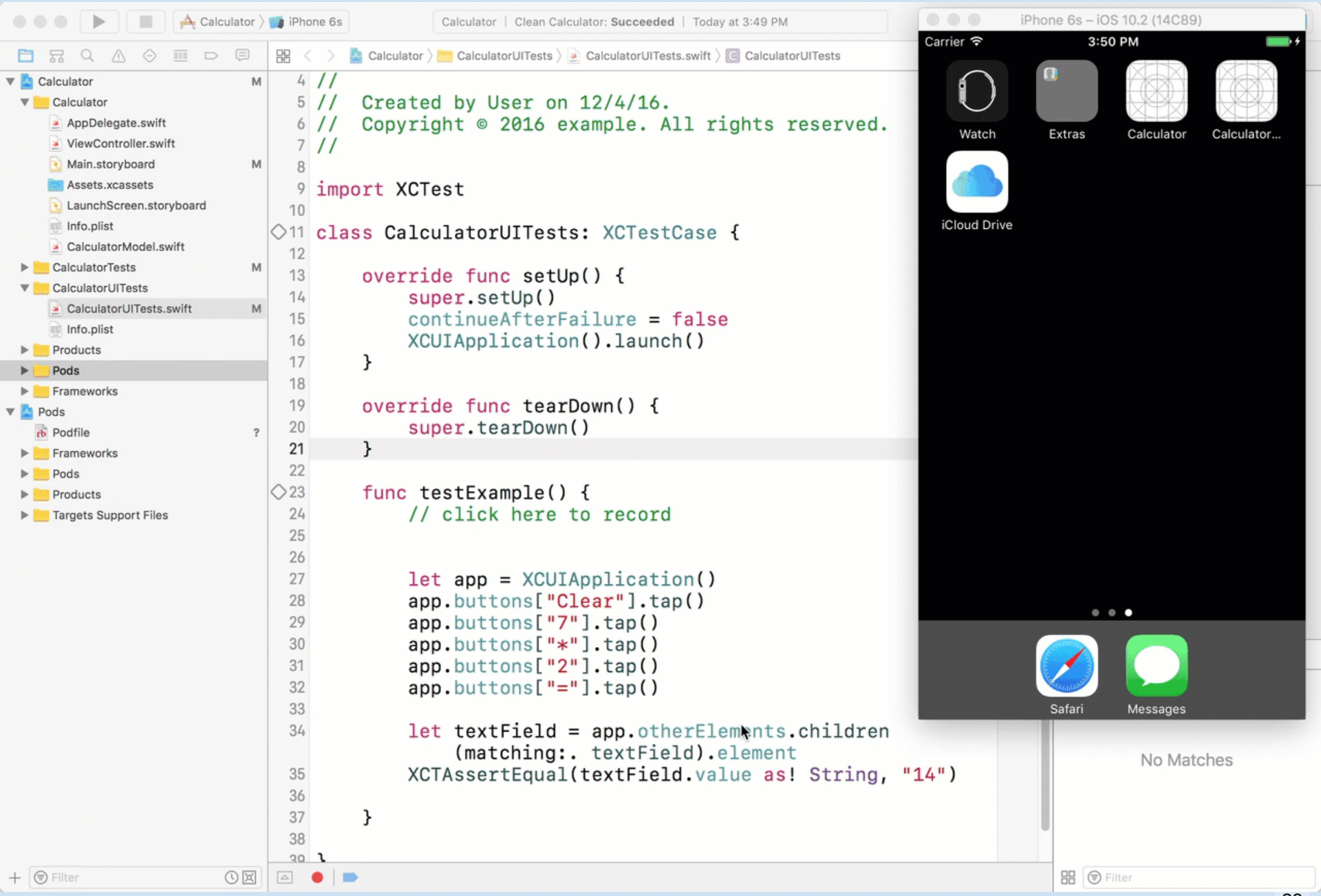
- User Interface Testing
- Two Options
  - Recorded Tests
  - Roll Your Own

Swift GUI Testing - XCUI

- XCUIApplication
  - Launch the app
- XCUIElementQuery
  - Find the XCUI Element to test
- XCUIElement
  - Perform Test

Swift GUI Testing - XCUI

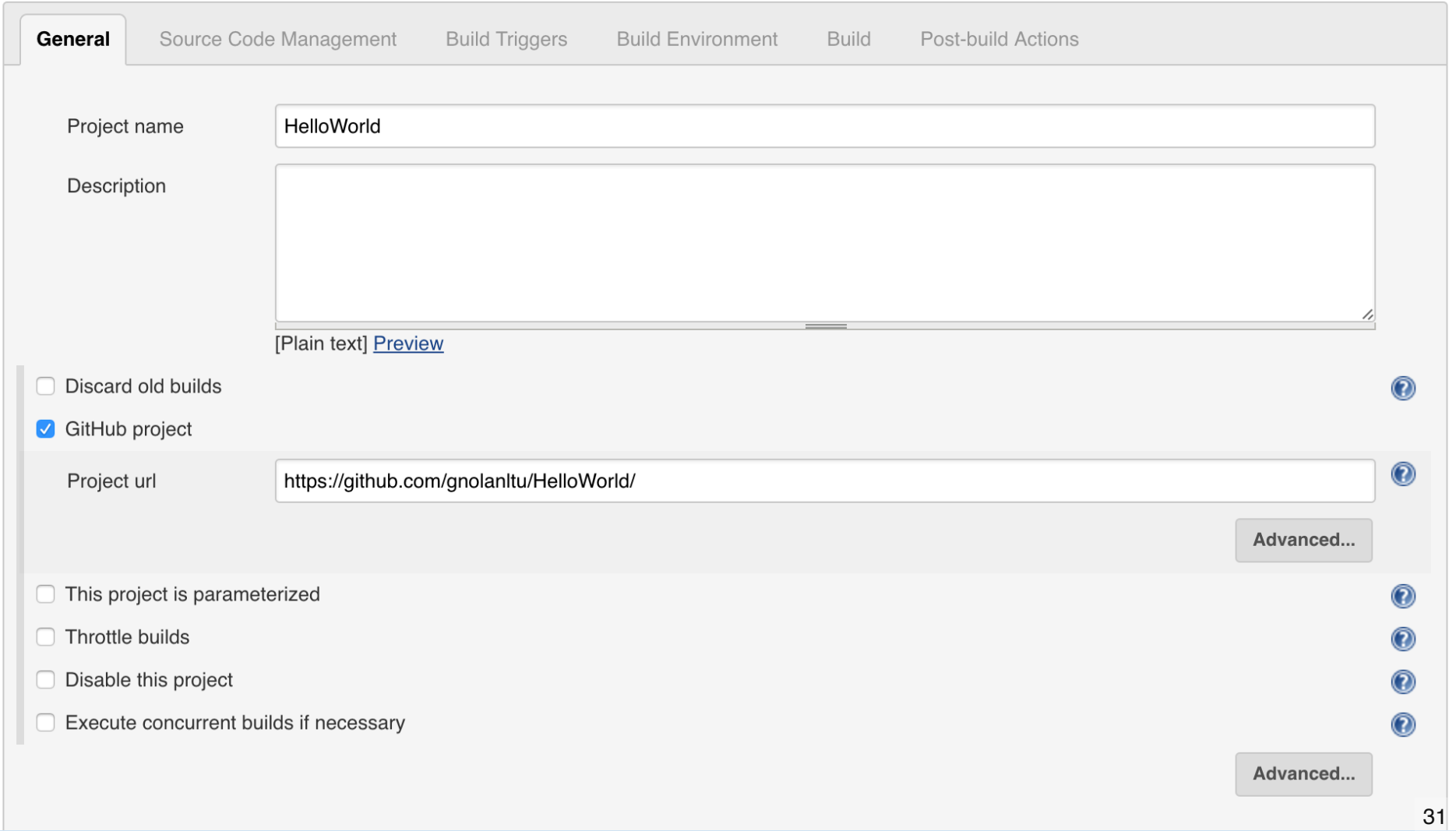



Calculator
iPhone 6s
Calculator | Clean Calculator: Succeeded | Today at 3:49 PM
Calculator
CalculatorUITests
CalculatorUITests.swift
CalculatorUITests
Calculator
Calculator
AppDelegate.swift
ViewController.swift
Main.storyboard
Assets.xcassets
LaunchScreen.storyboard
Info.plist
CalculatorModel.swift
CalculatorTests
CalculatorUITests
CalculatorUITests.swift
Info.plist
Products
Pods
Frameworks
Pods
Podfile
Frameworks
Pods
Products
Targets Support Files
Filter
//
// Created by User on 12/4/16.
// Copyright © 2016 example. All rights reserved.
//
import XCTest
class CalculatorUITests: XCTestCase {
override func setUp() {
super.setUp()
continueAfterFailure = false
XCUIApplication().launch()
}
override func tearDown() {
super.tearDown()
}
func testExample() {
// click here to record
let app = XCUIApplication()
app.buttons["Clear"].tap()
app.buttons["7"].tap()
app.buttons["*"].tap()
app.buttons["2"].tap()
app.buttons["="].tap()
let textField = app.otherElements.children
(matching:. textField).element
XCTAssertEqual(textField.value as! String, "14")
}
}
iPhone 6s – iOS 10.2 (14C89)
Carrier
3:50 PM
Watch
Extras
Calculator
Calculator...
iCloud Drive
Safari
Messages
No Matches
Filter

Swift GUI Testing Lab

1. Open Calculator project in xcode
2. Go to GUI Tests
3. Create new test file
4. record tests
5. Add assertion
6. Replay tests

Putting It All Together



Putting It All Together

www.cimgf.com/2015/05/26/setting-up-jenkins-ci-on-a-mac-2/

COCOA IS MY GIRLFRIEND
CIMGF.COM

Search...

About
Cocoa Code Snippets
ZDS Code Style Guide

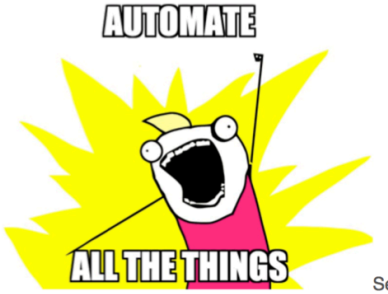
Categories
Advanced
Announcement
App Distribution
Apple
AppStore
Audio
Beginner
Blogging
Cocoa
Cocoa Touch
Coding Practice
Community
Continuous Integration
Core Animation
Core Data
Core Video
Development Environment
Frameworks

SETTING UP JENKINS CI ON A MAC

by Eric Cerney

26 MAY 2015

Continuous Integration



Setting up a CI server can be nothing but pain and suffering, yet many of us take on this challenge to save time in the long run. Some things I would rather do than set up a CI server:

1. Get a root canal at my desk waiting for Xcode SourceKit crashes to calm down.
2. Submit an Apple Radar for one of my many Xcode crashes.
3. Write a tutorial on how to set up a Jenkins CI server. (Oh hey!)

Whether you're being a good developer and doing your unit testing, are tired of co-workers breaking your code, or have clients that can't live without the latest and greatest build, CI will be your new best friend and butler.

There are tons of posts floating around the internet with bits and pieces on this topic; but none were comprehensive enough for me to get a server up and running. Because of this, I decided to write a thorough guide that will walk you through setting up a Jenkins CI Server on your machine of choice. In this instance we used a Mac Mini.

Note that our end goal with Jenkins was to build, test, and distribute apps for both the iOS and Android platforms. Some of the setup in this guide is aimed at accomplishing this goal.

Putting It All Together

1. Download the Jenkins Mac OS X native package from http://jenkins-ci.org.
2. Double click the .pkg file to install Jenkins.
3. Once done, your browser will open to http://localhost:8080 where Jenkins lives.
4. Make the Jenkins user an admin:
   ```
   sudo dseditgroup -o edit -a jenkins -t user admin
   ```
5. Add the Jenkins user to the developer group:
   ```
   sudo dscl . append /Groups/_developer GroupMembership jenkins
   ```
6. Make the Jenkins user automatically login when the computer is restarted
7. unload Jenkins as a Daemon:
   ```
   sudo launchctl unload /Library/LaunchDaemons/org.jenkins-ci.plist
   ```
8. move the .plist file, which defines how Jenkins will run, to the LaunchAgents folder:
   ```
   sudo mv /Library/LaunchDaemons/org.jenkins-ci.plist /Library/LaunchAgents/
   ```
9. Edit the plist file:
   ```
   sudo vim /Library/LaunchAgents/org.jenkins-ci.plist
   /* Remove the following lines */
   <key>SessionCreate</key
   <true />
   ```
10. reload the Launch Agent to restart Jenkins:
    ```
    sudo launchctl load /Library/LaunchAgents/org.jenkins-ci.plist
    ```

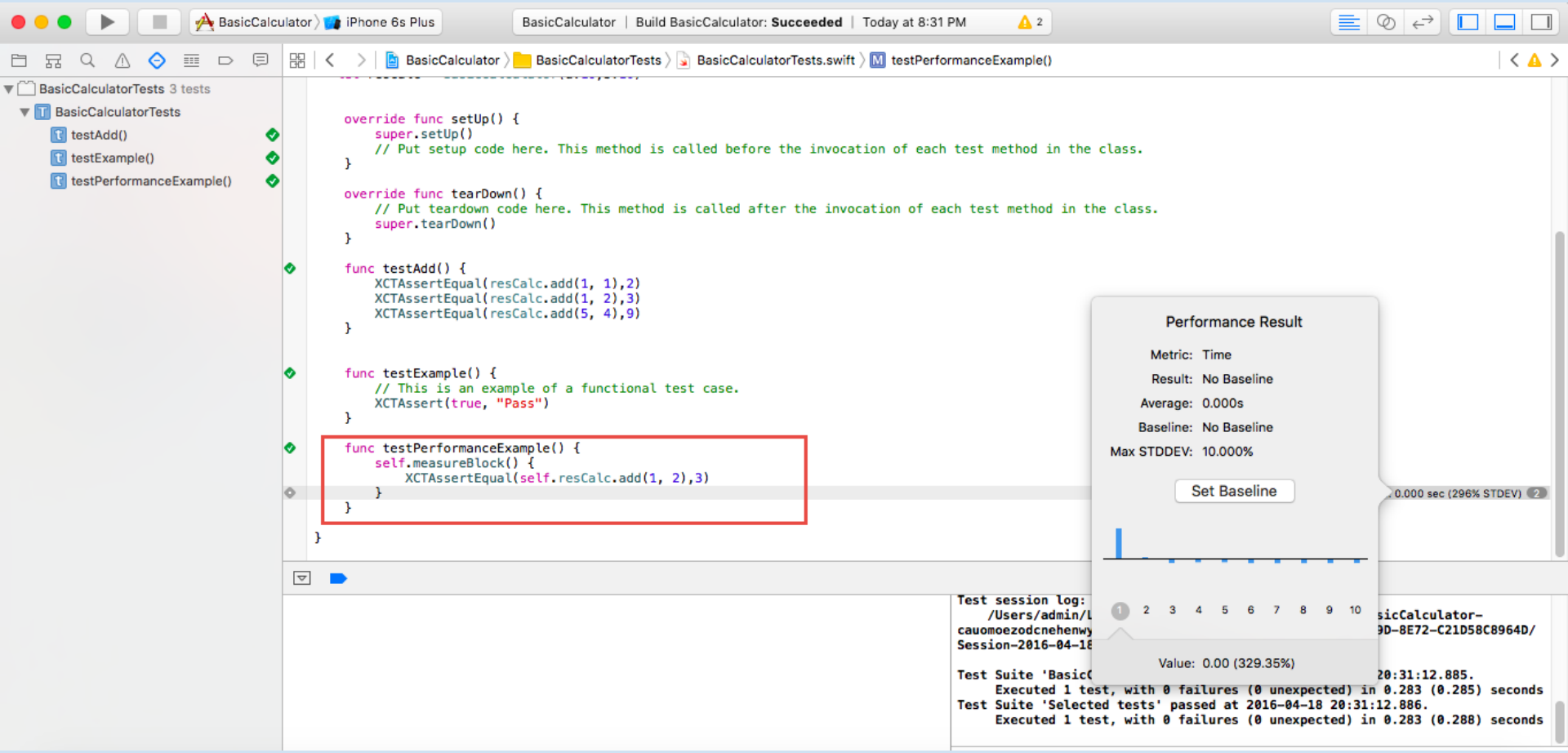
FIRST Principles

- F(ast)
- I(solated)
- R(epeatable)
- S(elf-verifying)
- T(imely) i.e. TDD not TAD

FIRST Principles - Fast

Feature	Small	Medium	Large
Network access	No	localhost only	Yes
Database	No	Yes	Yes
File system access	No	Yes	Yes
Use external systems	No	Discouraged	Yes
Multiple threads	No	Yes	Yes
Sleep statements	No	Yes	Yes
System properties	No	Yes	Yes
Time limit (seconds)	60	300	900+

FIRST Principles - Fast



Performance Lab

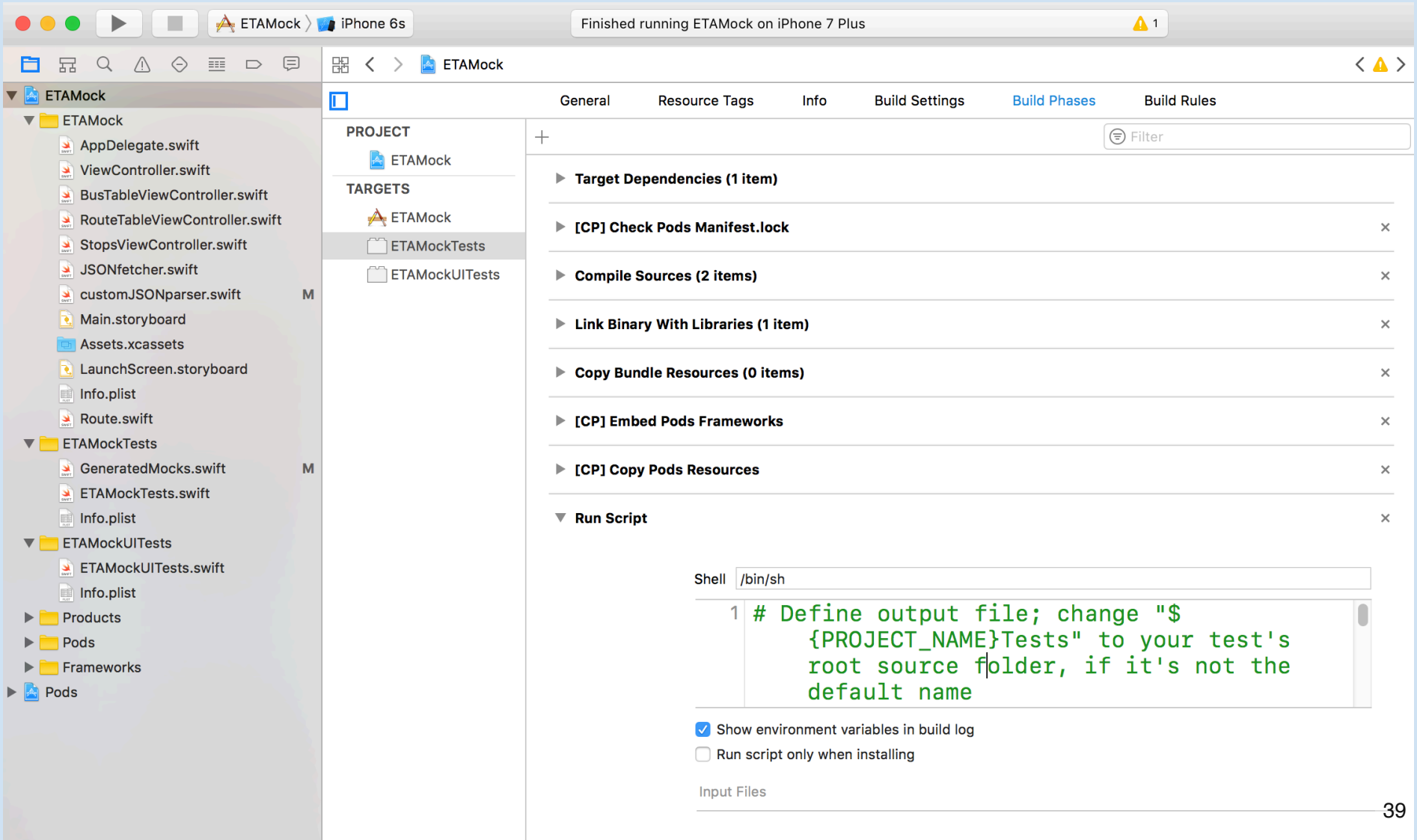
```
1. Open Calculator project
2. Create new test file
3. Add assertion in Performance section
4. Test and measure speed.
```

FIRST Principles - Isolated

```
when(methodIsCalled).thenReturn(aValue);
```

```
// Cuckoo

stub(mock) { stub in
  when(stub.readWriteProperty.get).thenReturn(10)
}
```

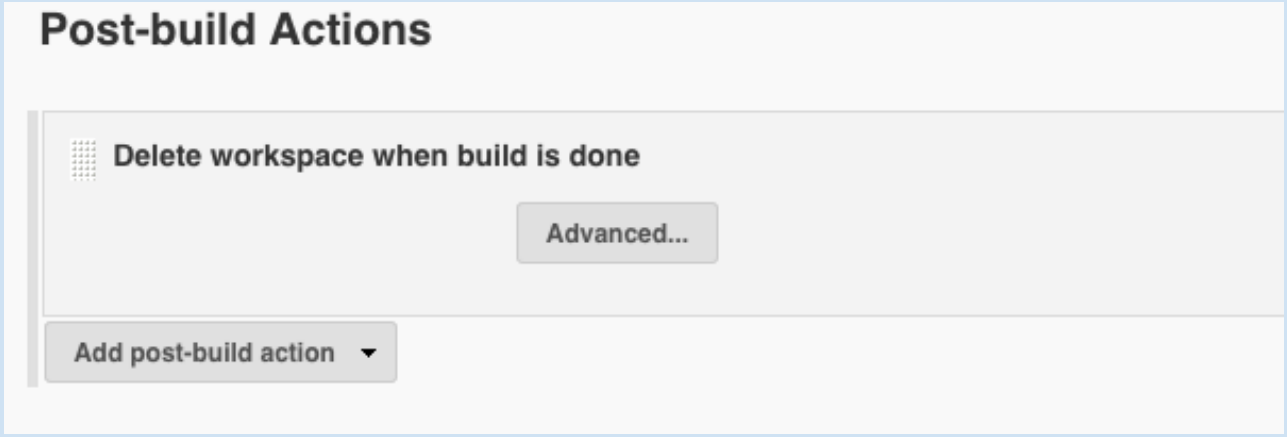
Isolated - Cuckoo



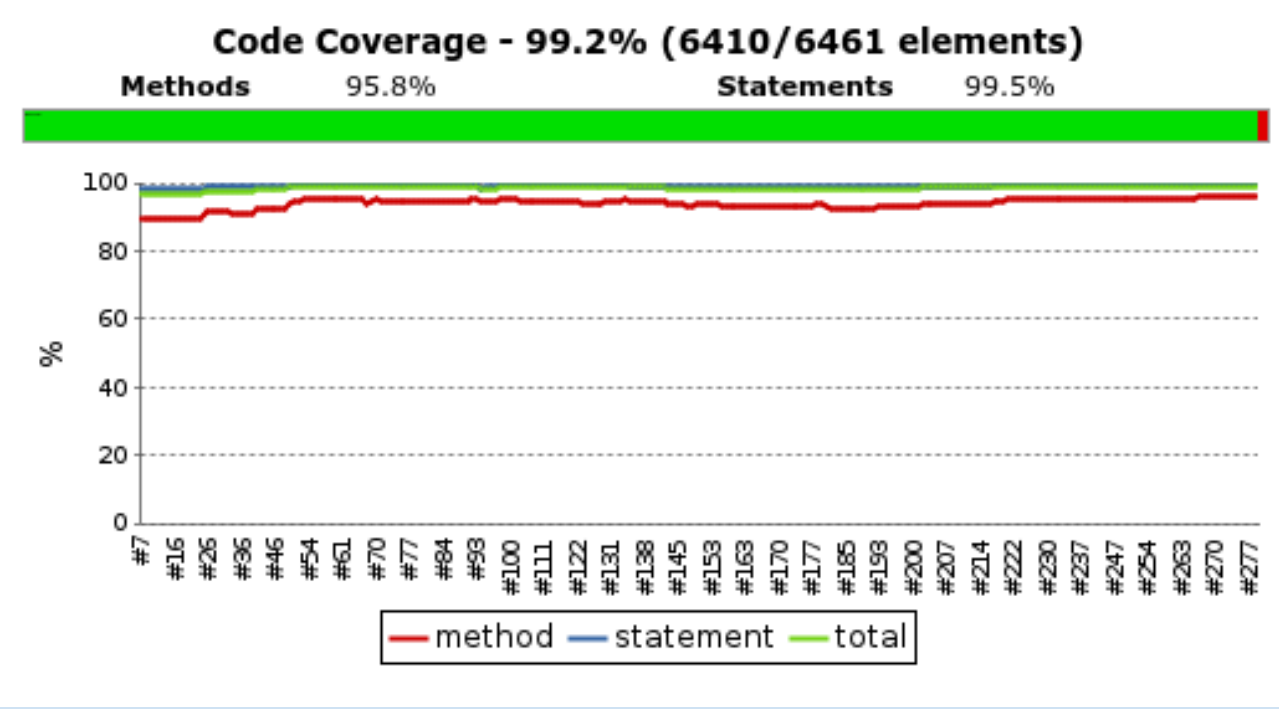
Isolated - Cuckoo

```
# Define output file; change "${PROJECT_NAME}Tests" to your test's
root source folder, if it's not the default name

OUTPUT_FILE="./${PROJECT_NAME}Tests/GeneratedMocks.swift"
echo "Generated Mocks File = ${OUTPUT_FILE}"

# Define input directory; change "${PROJECT_NAME}" to your project's root
source folder, if it's not the default name
INPUT_DIR="./${PROJECT_NAME}"
echo "Mocks Input Directory = ${INPUT_DIR}"

# Generate mock files; include as many input files as you'd like to create mocks for
${PODS_ROOT}/Cuckoo/run generate --testable "${PROJECT_NAME}" \
--output "${OUTPUT_FILE}" \
"${INPUT_DIR}/FileName1.swift" \
"${INPUT_DIR}/FileName2.swift" \
"${INPUT_DIR}/FileName3.swift"
# ... and so forth
```

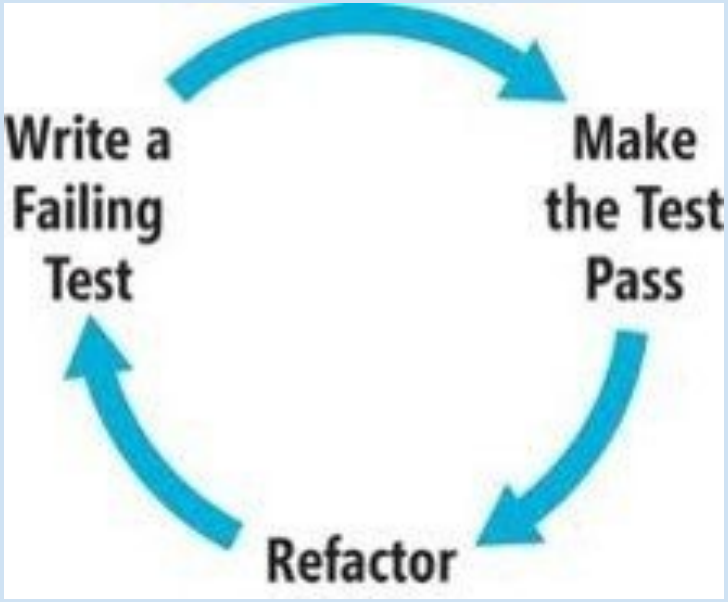
FIRST Principle - Repeatable



FIRST Prin - Self-Verifying



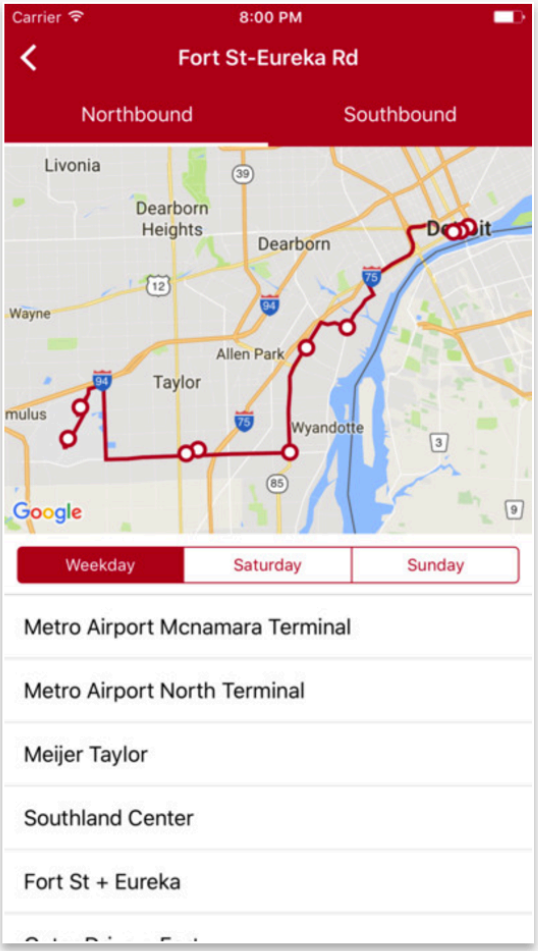
FIRST Principles - Timely

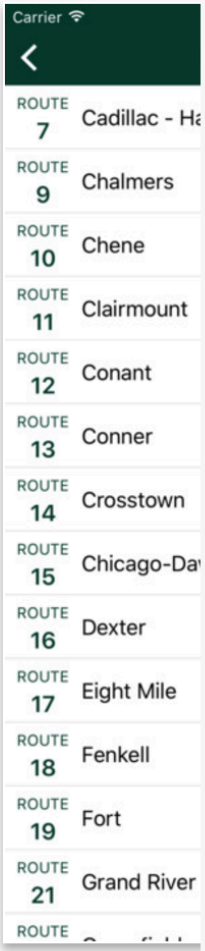


Sample App

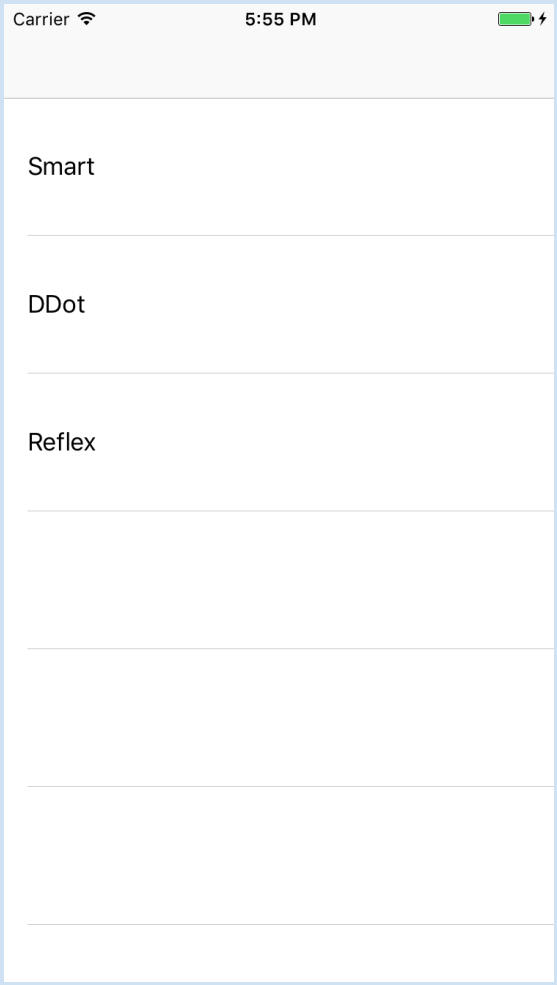
Screenshots

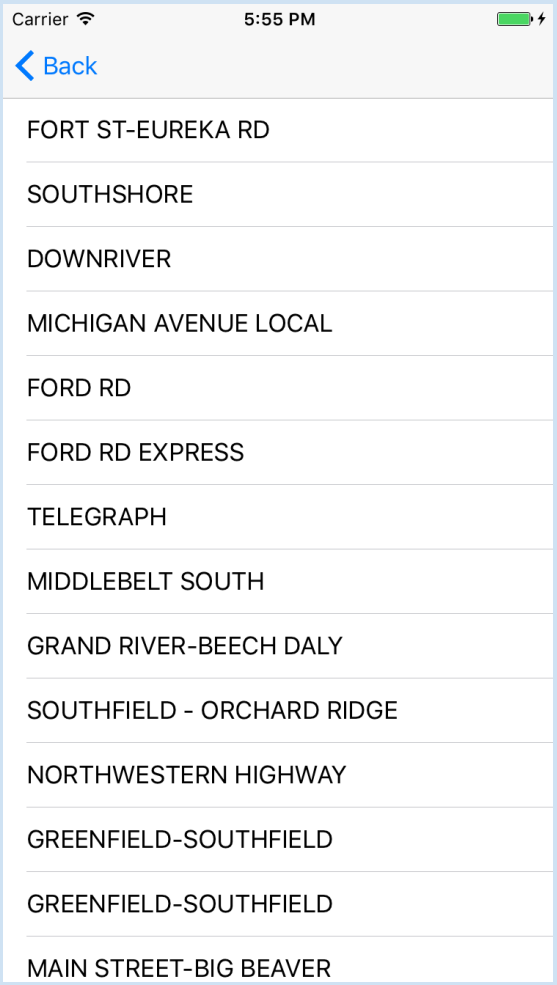
iPhone | iPad

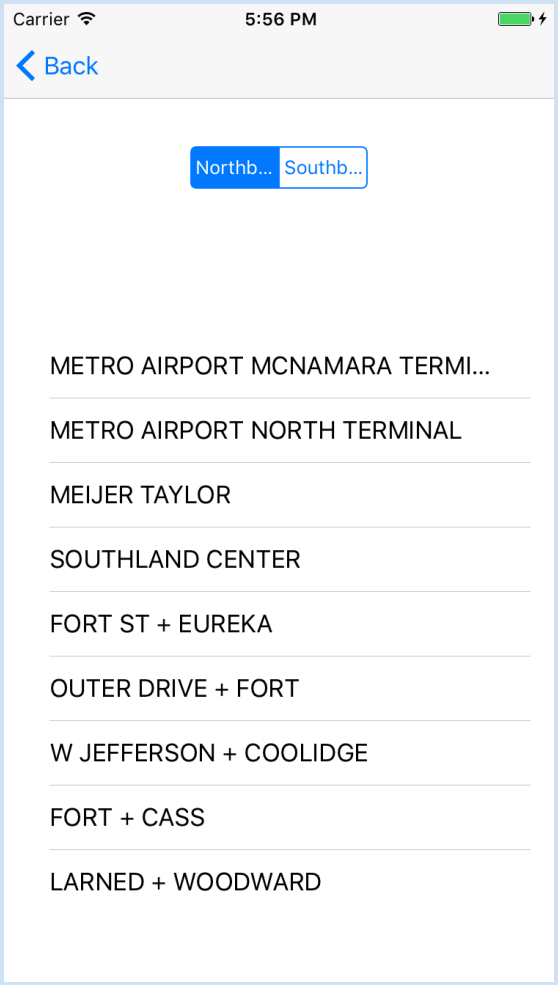




Sample App


Carrier
5:55 PM
Smart
DDot
Reflex


Carrier
5:55 PM
Back
FORT ST-EUREKA RD
SOUTHSHORE
DOWNRIVER
MICHIGAN AVENUE LOCAL
FORD RD
FORD RD EXPRESS
TELEGRAPH
MIDDLEBELT SOUTH
GRAND RIVER-BEECH DALY
SOUTHFIELD - ORCHARD RIDGE
NORTHWESTERN HIGHWAY
GREENFIELD-SOUTHFIELD
GREENFIELD-SOUTHFIELD
MAIN STREET-BIG BEAVER


Carrier
5:56 PM
Back
Northb...
Southb...
METRO AIRPORT MCNAMARA TERMI...
METRO AIRPORT NORTH TERMINAL
MEIJER TAYLOR
SOUTHLAND CENTER
FORT ST + EUREKA
OUTER DRIVE + FORT
W JEFFERSON + COOLIDGE
FORT + CASS
LARNED + WOODWARD

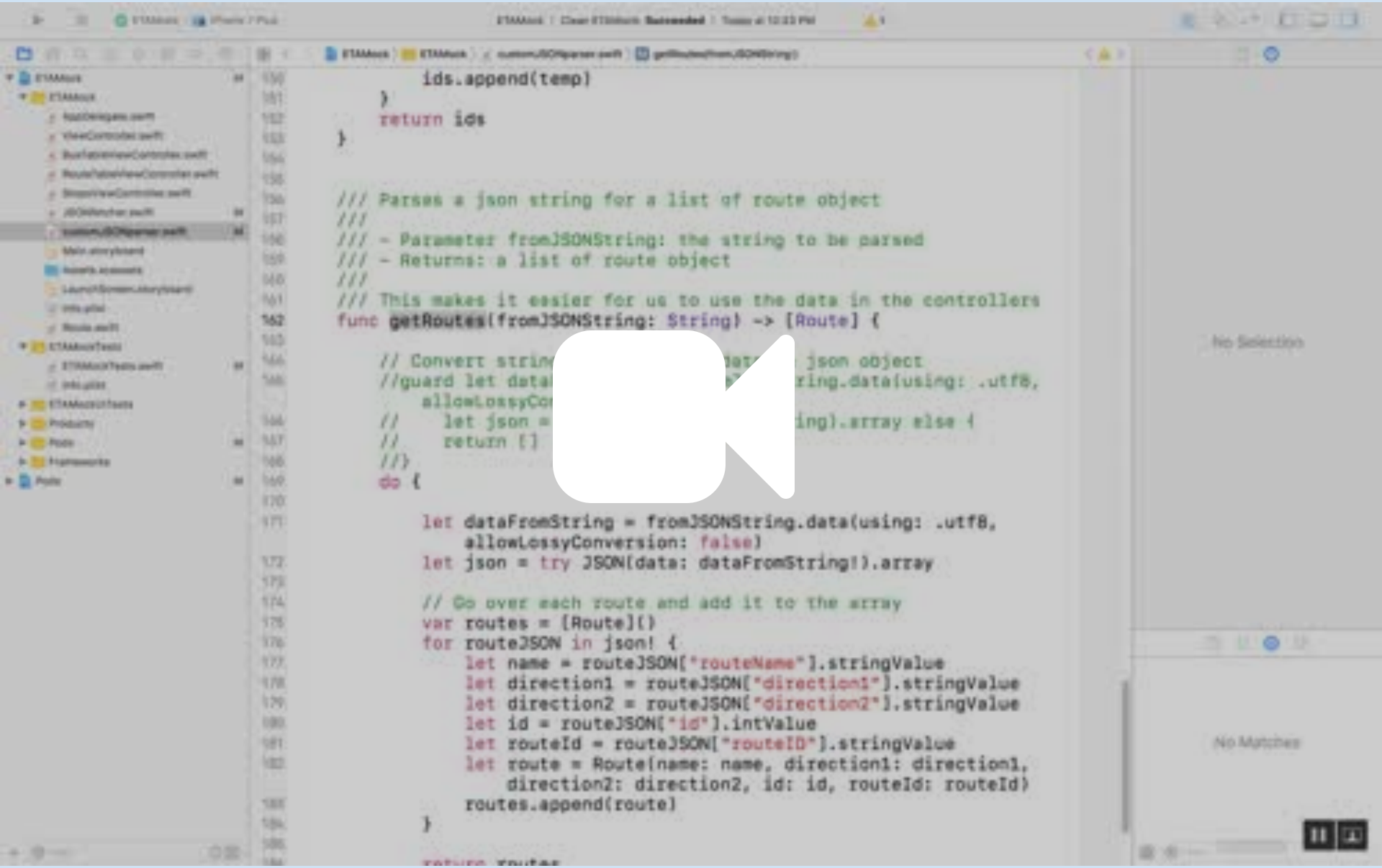
Sample App - Unit Test

```
func testParseRoutes() {
    let mock = MockJSONfetcher()

    stub(mock) { mock in
        when(mock.callApi(url: any(), completion: anyClosure())).then { url, closure in
            closure(self.testRouteJson)
        }
    }

    mock.callApi(url: url) { data in
        XCTAssertEqual(data, self.testRouteJson)
        let parser = customJSONparser(companyIndex: 1)
        let route = Route(name: "FORT ST-EUREKA RD",
                direction1: "Northbound",
                    direction2: "Southbound", id: 1, routeId: "125")
        XCTAssertEqual(parser.getRoutes(fromJSONString: data), [route])
    }
}
```

```
xcodebuild test -workspace ETAMock.xcworkspace -scheme ETAMock
-destination 'platform=iOS Simulator, name=iPhone 7 Plus'
```

Sample App - Unit Test

```
# Define output file; change "${PROJECT_NAME}Tests" to your test's root source folder, if it's not the default name
OUTPUT_FILE="./${PROJECT_NAME}Tests/GeneratedMocks.swift"
echo "Generated Mocks File = ${OUTPUT_FILE}"

# Define input directory; change "${PROJECT_NAME}" to your project's root source folder, if it's not the default name
INPUT_DIR="./${PROJECT_NAME}"
echo "Mocks Input Directory = ${INPUT_DIR}"

# Generate mock files; include as many input files as you'd like to create mocks for
${PODS_ROOT}/Cuckoo/run generate --testable "${PROJECT_NAME}" \
--output "${OUTPUT_FILE}" \
"${INPUT_DIR}/JSONFetcher.swift"

# ... and so forth

# After running once, locate `GeneratedMocks.swift` and drag it into your Xcode test target group
```

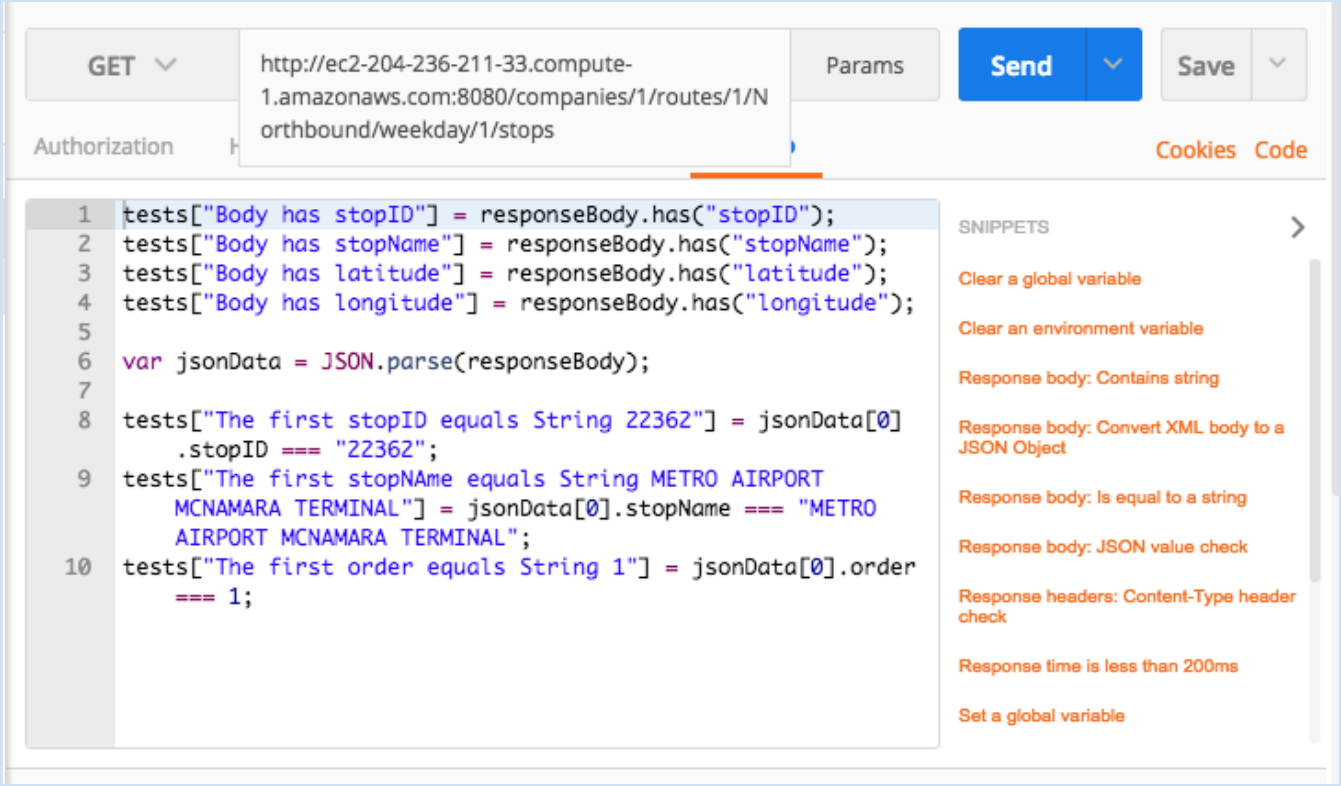

ids.append(temp)
}
return ids
}
/// Parses a json string for a list of route object
///
/// - Parameter fromJSONString: the string to be parsed
/// - Returns: a list of route object
///
/// This makes it easier for us to use the data in the controllers
func getRoutes(fromJSONString: String) -> [Route] {
// Convert strin
//guard let data
allowLossyCo
// let json =
// return []
//}
do {
let dataFromString = fromJSONString.data(using: .utf8,
allowLossyConversion: false)
let json = try JSON(data: dataFromString!).array
// Go over each route and add it to the array
var routes = [Route]()
for routeJSON in json! {
let name = routeJSON["routeName"].stringValue
let direction1 = routeJSON["direction1"].stringValue
let direction2 = routeJSON["direction2"].stringValue
let id = routeJSON["id"].intValue
let routeId = routeJSON["routeID"].stringValue
let route = Route(name: name, direction1: direction1,
direction2: direction2, id: id, routeId: routeId)
routes.append(route)
}
No Selection
No Matches

Isolated - Cuckoo Lab

0. sudo gem install cocoapods
1. git clone http://github.com/gnolanltu/ETAMock
2. View the podfile and see pod “Cuckoo” as a test target.
3. Run pod install.
4. Close the project and reopen the workspace.
5. Click on the project folder then choose Test Target ➤ Build Phases.
6. Click + and choose New Run Script Phase.
7. Add Listing to the Run Script section, making sure to modify the input files that you want to mock.
8. Build the project.
9. Run the tests.
10. Drag and drop GeneratedMocks.swift into the test section.
11. Uncomment out your mocked tests
12. Run the mocked tests.

Sample App - API Test

Sample App - API Test




```
newman run ETAJson.postman_collection
```

Sample App - API Test

```
RIIS-MBPRO-13:Android User$ newman run ETAJson.postman_collection.json
newman

ETAJson

→ http://ec2-204-236-211-33.compute-1.amazonaws.com:8080/companies/1/routes/1/Northbound/weekday/1/stops
  GET http://ec2-204-236-211-33.compute-1.amazonaws.com:8080/companies/1/routes/1/Northbound/weekday/1/stops [200 OK, 1.74KB,
 103ms]
  ✓  Body has stopID
  ✓  Body has stopName
  ✓  Body has latitude
  ✓  Body has longitude
  ✓  The first stopID equals String 22362
  ✓  The first stopNAme equals String METRO AIRPORT MCNAMARA TERMINAL
  ✓  The first order equals String 1
```

	executed	failed
iterations	1	0
requests	1	0
test-scripts	1	0
prerequest-scripts	0	0
assertions	7	0
total run duration: 152ms		
total data received: 1.59KB (approx)		
average response time: 103ms		

Sample App - GUI Test

```
    func testExample() {
        let app = XCUIApplication()
        let tablesQuery = app.tables
        tablesQuery.staticTexts["Smart"].tap()
        tablesQuery.staticTexts["SOUTHSHORE"].tap()
        app.buttons["Southbound"].tap()
        XCTAssert(tablesQuery.staticTexts["JEFFERSON + SOUTHFIELD"].exists)


        app.navigationBars["ETAMock.StopsView"].children(matching: .button)
                        .matching(identifier: "Back").element(boundBy: 0).tap()
        tablesQuery.staticTexts["MICHIGAN AVENUE LOCAL"].tap()
        app.buttons["Westbound"].tap()
        XCTAssert(tablesQuery.staticTexts["MICHIGAN + CASS"].exists)
    }
```

```
xcodebuild build -workspace ETAMock.xcworkspace -scheme ETAMock
-destination 'platform=iOS Simulator, name=iPhone 7 Plus'
```

Sample App - Jenkins

Build

Execute shell

Command

```
#!/bin/bash -l
pod install
```

See the list of available environment variables

Advanced...

Execute shell

Command

```
export PATH=/sbin:/usr/sbin:/bin:/usr/bin:/usr/local/bin

/usr/local/Cellar/node/7.8.0/libexec/npm/bin/newman run ETAJson.postman_collection.json
```

See the list of available environment variables

Advanced...

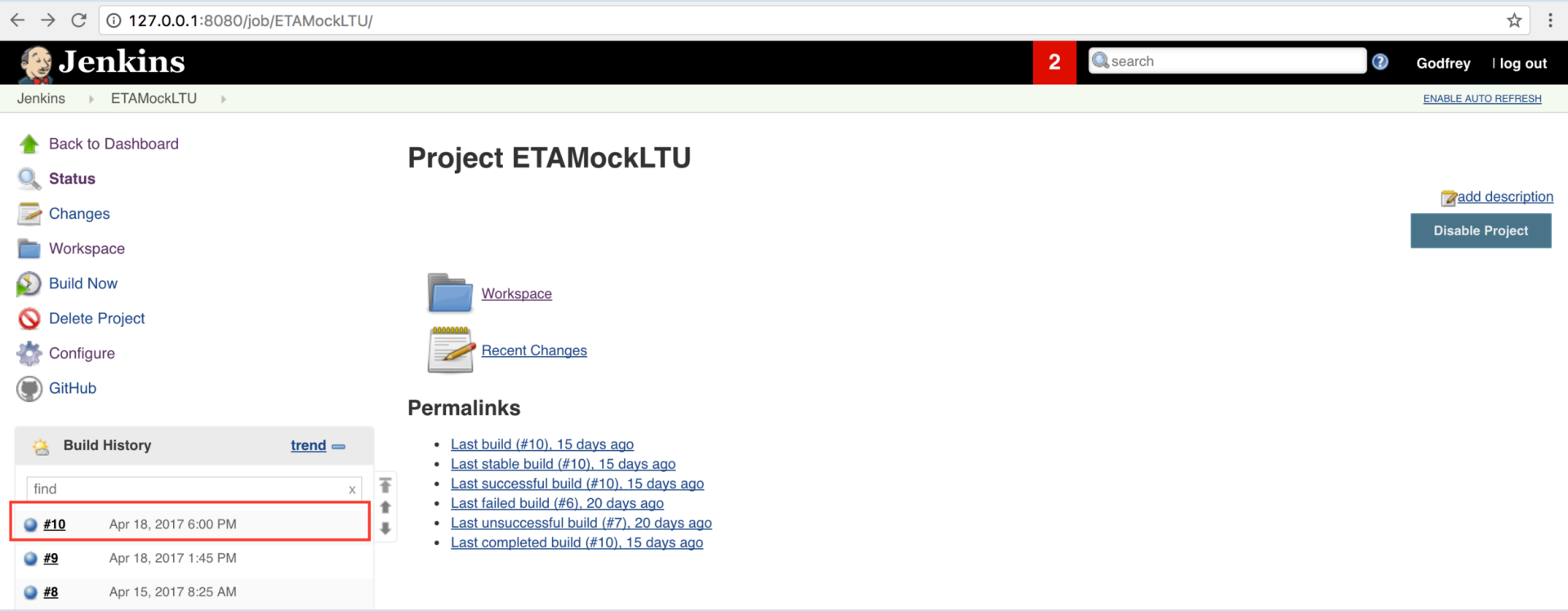
Execute shell

Command

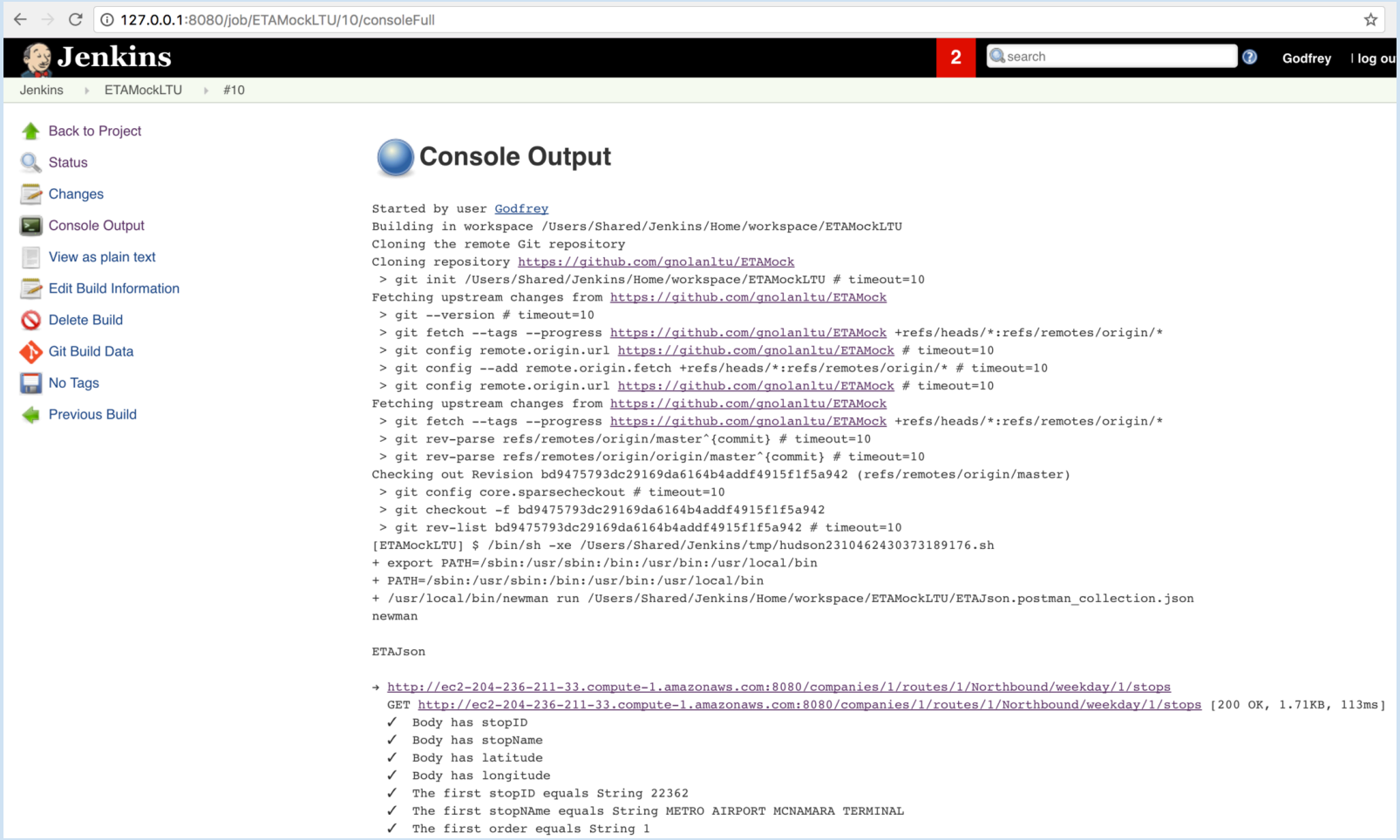
```
xcodebuild build -workspace ETAMock.xcworkspace -scheme ETAMock -destination 'platform=iOS Simulator,name=iPhone 7 Plus'
```

See the list of available environment variables

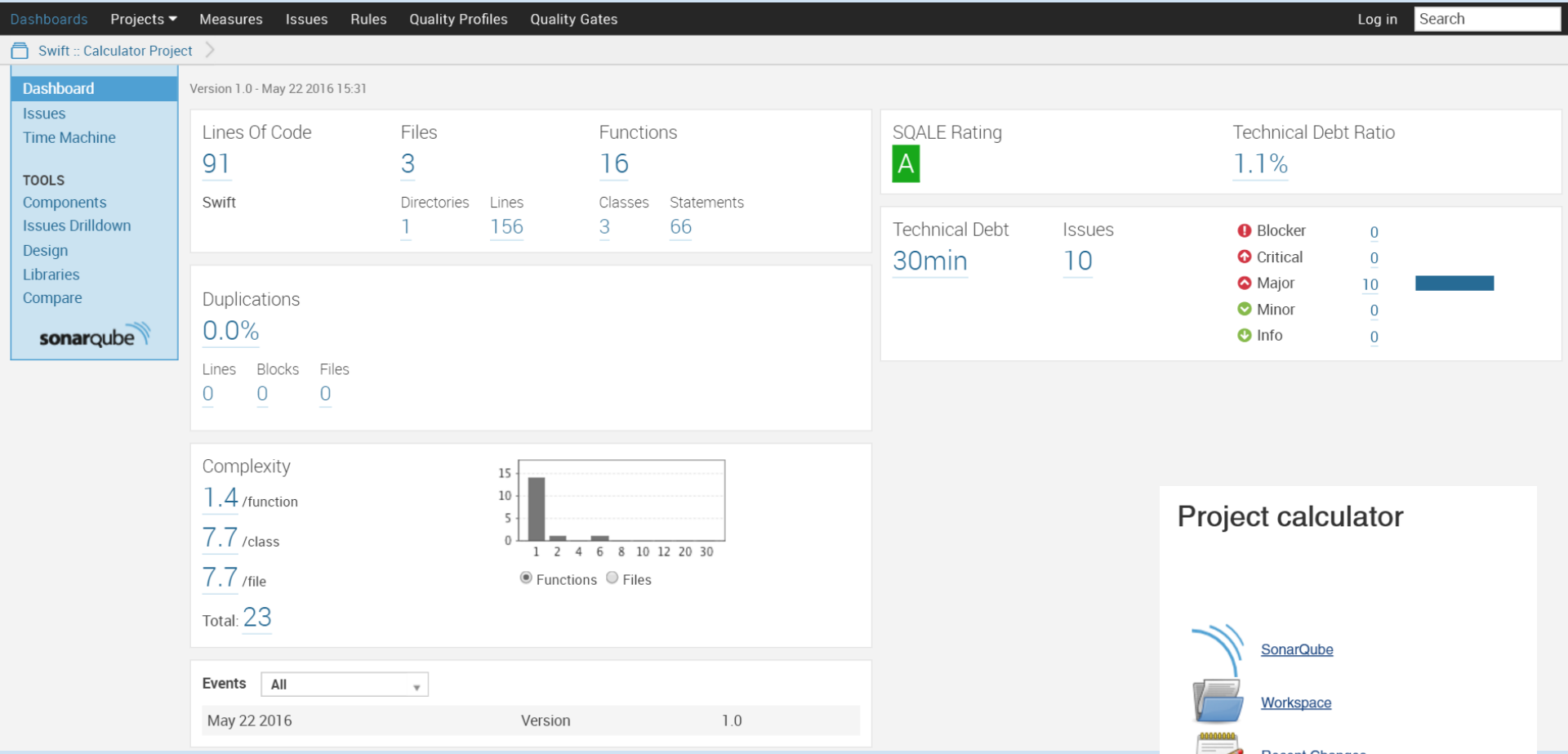
Sample App - Jenkins



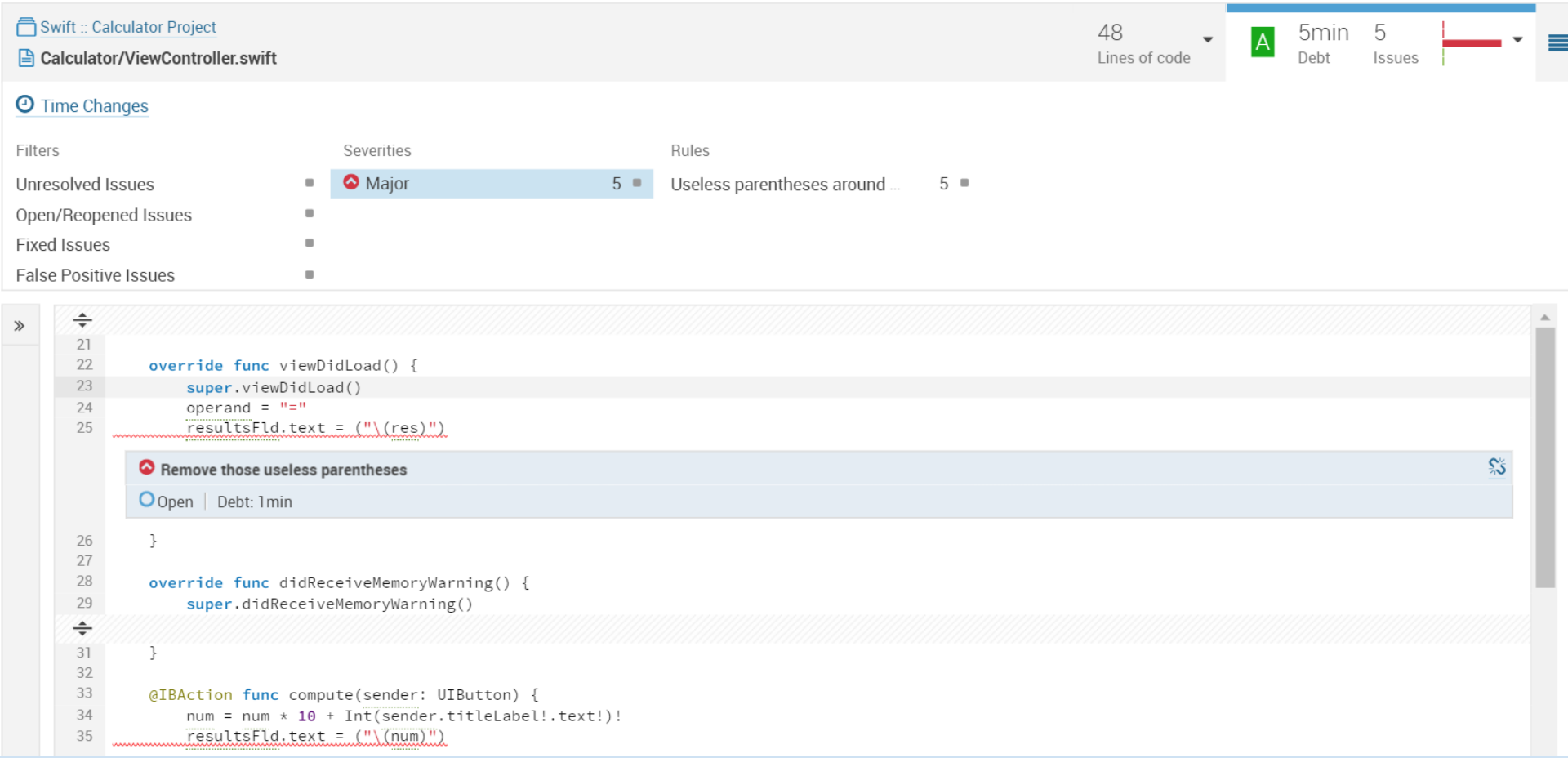
Sample App - Jenkins



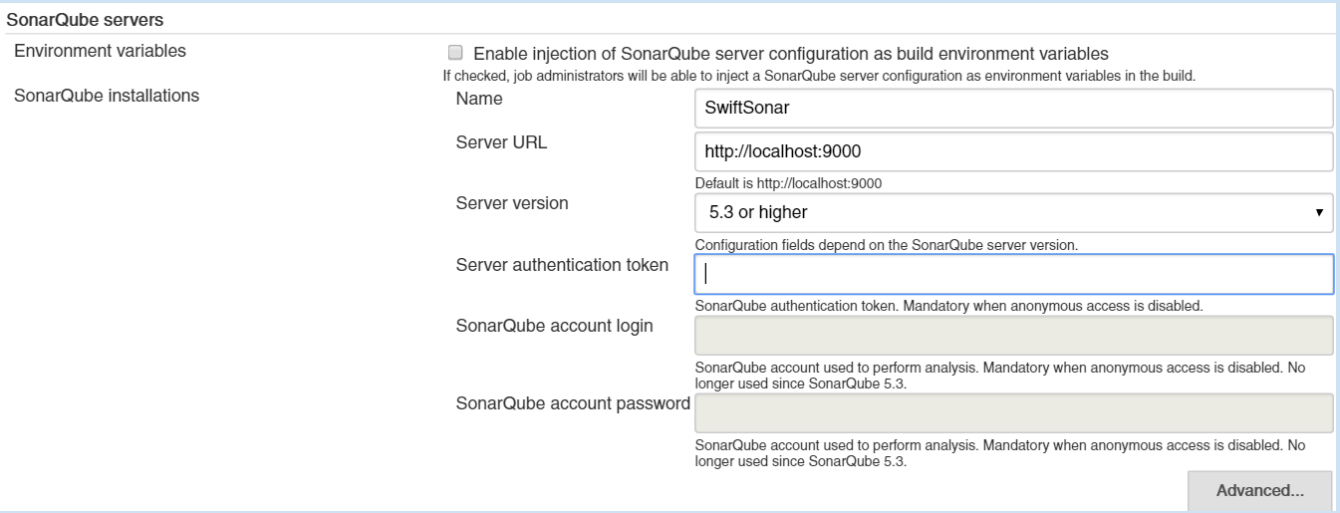
SonarQube - Jenkins

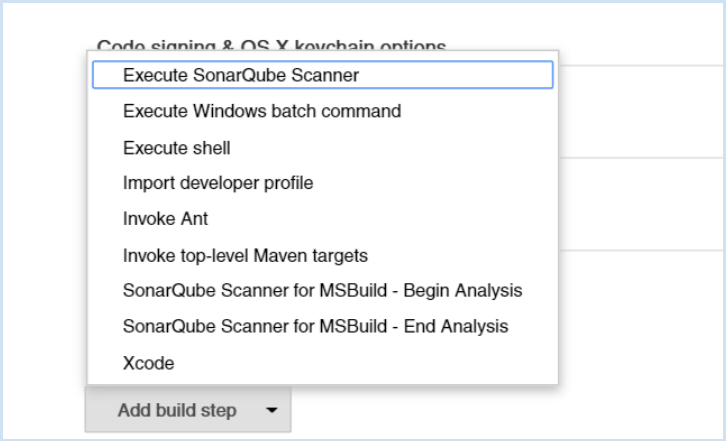


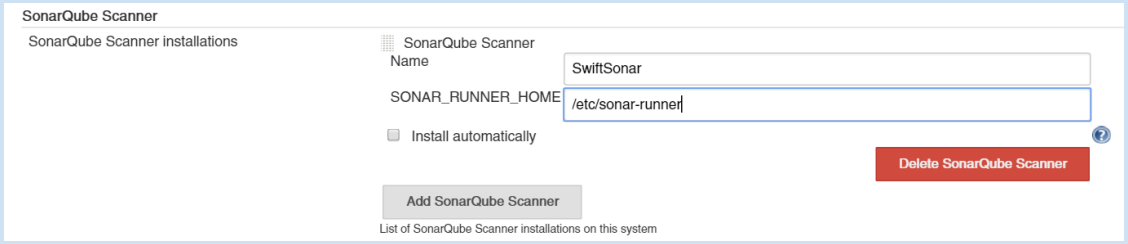
SonarQube - Jenkins



SonarQube - Jenkins







Fastlane

```
1. brew cask install fastlane
2. Run in SimpleETA project directory
    fastlane init
3. Answer questions
4. Run
    fastlane test
```

URLs

http://riis.com/blog

https://getpostman.com

https://github.com/postmanlabs/newman

https://github.com/Brightify/Cuckoo

https://jenkins.io

http://www.cimgf.com/2015/05/26/setting-up-jenkins-ci-on-a-mac-2/

https://github.com/gnolanltu/ETAMock

https://sonarqube.org

https://github.com/fastlane/fastlane

CONTACT INFO

godfrey@riis.com

@godfreynolan